The Prediction of Code Clone Quality Based on Bayesian Network

Dongrui Liu¹, Dongsheng Liu², Liping Zhang³, Min Hou⁴ and Chunhui Wang⁵

Inner Mongolia Normal University Hohhot, Inner Mongolia, China
nmlds@126.com

Abstract

This paper researched on the quality of code clone in the software, evaluated the code clone quality of the current versions. Then using Bayesian network to train the existing sample data to get the prediction model of code clone that is able to predict the quality. The prediction results are able to help developers decide which code clone should be reconstructed or efficiently reused. The experiment shows that the method can be used to predict the quality of code clone in software more accurately.

Keywords: Code clone; Bayesian network; Quality model; Prediction; Reconstruct

1. Introduction

With the continuous development of software architecture, software design patterns and related areas, people prefer reusing the existing achievement at all frameworks of software development rather than repeat what others have created in the past during the process of developing new software systems [1]. Lots of code clone [2], also called clones, are generated in the process of software reuse. The clones cause both positive and negative influence to the development and maintenance. It is not only reduce the potential risks of writing new codes but also improve the development efficiency to copy a section of high quality codes those are beneficial clones [3]. However, It is possible to introduce new Bugs when copy a section of low quality codes [3].

The early research focuses on finding out the clones in the software and removing them by reconstruction technique [4]. In recent years, a number of studies have found that some of the clones, should be packaged to the reusable construction that must have quality assurance for the reuse of software system, not necessarily to reconstruct or eliminate [5], is conducive to the development and maintenance [6]. There has been an urgent need for the method of evaluating the quality of the clones to provide effective protection for software reuse and management.

2. Related Technology

Clone management [7] is the method adopts a variety of techniques to reuse some high quality clones effectively and try to avoid the low quality clones, make full use of the positive role of the clone code and reduce its negative effect. In the process of clone management includes numbers of technology, for instance: clone reconstruction, clone quality evaluation, machine learning method and its like.

2.1. Clone Reconstruction

Clone quality has a close relationship with clone reconstruction [8]. Mondal Manishankar et. al., have defined a particular clone change pattern, the Similarity Preserving Change Pattern (SPCP), and considered the cloned fragments that changed according to this pattern (i.e., the SPCP clones) as important candidates for refactoring [9].
Fowler et al., [10] introduced a catalog of 72 refactoring patterns, and till date, the number has increased to 93 [11]. According to the earlier studies [12] Extract Method Refactoring, Remove Method Refactoring, Pull-Up Method Refactoring, Parameterization Refactoring and Template Methods Refactoring are the most commonly advocated refactoring techniques to remove cloned codes. The details of the models can be found in the refactoring and the other studies [13].

2.2. The Quality Evaluation of Code Clone

Numerous studies have found that many clones are advantageous to the software development and maintenance. It is not only reduce the potential risk of writing new code, but also improve the development efficiency to copy a section of the source code without defects, the code called beneficial clone.

Weng Xiumu et al., [14] built a code quality model to evaluate the code quality by maintainability, testability, readability, portability, reusability and its like. But the study did not measure the usefulness of these factors. Gong Lina [15] et al., proposed a rough set based fuzzy neural network algorithm, choosed machine learning method to express the relationship between the quality of the software and the uncertainty of the software, to evaluate software quality. But the study did not have a depth study on the use of the evaluation metrics. Radhika D et al., evaluated the quality of clones by Evaluation Method for Internal Software Quality (EMISQ) [16], provided important reconstruction suggestion to developers from three factors: maintenance overhead for cloned code (MO), lack of software quality of the clones (LSQ) and refactoring magnitude (RFT), and discuss various criteria that help in prioritizing the clone results [17]. The studies above have achieved corrective management [7] but preventive management [7].

2.3. Bayesian Network

There have been many machine learning methods. Back Propagation neural network, Bayesian network, Decision Tree and Support Vector Machine (SVM) are the current mainstream and have a wide range of data mining applications [18]. Huang Jianming have inference and prediction to course score using Bayesian network [19]. Luo Ling et al., put forward a learning style model based on TAN Bayesian network learning according to the presupposed learning style, which can realize the automatic detection of students learning style by mining these data come from students’ network study behavior [20].

3. Prediction of Code Clone Quality

In this paper the combination of the Bayesian network method and the quality evaluation of the code clone was proposed and we built a method to predict the quality of code clone described at great length in the following section.

3.1. Evaluation Method for Code Clone

The evaluation method for software quality was borrowed to evaluate the quality of clones extracting metrics to evaluate the quality of clones. Radhika D et al., provided important reconstruction suggestion based on EMISQ to developers from three factors: MO, LSQ and RFT. But the research was very difficult to evaluate the quality of code clone accurately without the characteristics of clone code. And RFT was not considered because of the reconstruction of clone was difficult to achieve automatically.

In this paper the metrics were extracted from MO and LSQ to evaluate the quality of clones. And the metrics of Refac was determined by both MO and LSQ. Refac which called the metric of clone reconstruction provided the reconstruction and encapsulation of the code clone. For example when the LSQ of the clone is high and the MO is low Refac is considered Y, means the clone needs to be reconstructed. And according to our
previous research on clone harmfulness, the clone of inconsistent change is likely to cause a large number bugs’ propagation during software evolution. So the inconsistent metrics were considered. The following are the major description.

3.1.1. Maintenance Overhead for Code Clone (MO):

(1) Lines of clone (LC): This indicates number of cloned lines in a clone fragment. Larger lines indicates higher maintenance. LC was extracted by our detection tools CCFinder [21].

(2) Clone Change Frequency (CCF): Larger frequency means larger probability of inconsistent change of clone [22]. According to the previous researches the key index of Bugs in the software was caused by the inconsistent change during the process of clone evolution [22].

(3) Days since Last Clone Change (DLCC): The heuristic followed is more recently a clone has been changed, more likely it is to be changed again. DLCC had an influence on the inconsistent change.

(4) Clone Age (CA): It has been suggested that the longer lived the clone was, the more stable it was and hence had lower maintenance overhead.

(5) Clone Complexity (CC): The higher the CC, extracted by SourceMonitor [23], was the more the MO cost.

Calculate the MO value of each clone group by the above metric values. The value for each is within the range [0, 1] and is dependent on the maximum absolute value of the metric. Normalized MO value\(^{\text{OM}}(\text{C}(n, j))\) is expressed as in Eq. (1).

\[
\text{MO}(\text{C}(n, j)) = \sum_{i=1}^{\text{Max}} \left| \frac{\text{CC}(\text{C}(n, j))}{\text{Max}(\text{CC}(C(n, j)))} \right|
\]

\[
\text{C}(n, j) \text{ indicates the No. j clone group in the No. n software version. The MO of individual clone fragment can then be calculated as a sum of these normalized metrics’ values expressed as in Eq. (2).}

\[
\text{LSQ}(\text{C}(n, j)) = \sum_{i=1}^{\text{Max}} \left| \frac{\text{C}(\text{C}(n, j))}{\text{Max}(\text{C}(n, j))} \right|
\]

3.1.2. Lack of Software Quality of the Clone (LSQ): LSQ was determined by different static analysis rules, and then determined the possibility of cloning code bug. FxCop, Gendarme and FlawFinder were used to extract the values of static analysis rules as following.

(1) Message Level (Leveli): The measure values from Level1 to Level5 based on Common Weakness Enumeration (CWE) [24]. Level1 caused a low probability of bugs while Level5 was the most likely to cause bugs. The Level value of the clone code in the C software was detected by FlawFinder [25].

(2) Weight of the Rules (Wi): The value ranges from 0 to 1. According to the severity of Bugs induced by Level, the corresponding weight was added. Different software added different weight values based on expertise.

(3) Criticality of the Rule (Crit): The metric, analysed by both Leveli and Wi, determined the importance of Level.

(4) Count of Violations of a Clone (CoV).

The value of LSQ was determined by the rules above expressed as in Eq. (3).
$$LSQ(C_{(n,j)}) = \frac{\sum_{i=1}^{j} Crit_i \cdot CoV_{(n,j)}}{\sum_{i=1}^{j} CoV_{(n,j)}}$$

$$\sum_{i=1}^{j} CoV_{(n,j)}$$ means the account of the rules number in the No. n software version of No j clone group, $$\sum_{i=1}^{j} Crit_i \cdot CoV_{(n,j)}$$ indicates the weighted sum of the clones that violate the static rules.

Given all that we extracted and calculated 14 metric values, shown in Table 1, from two aspects of LSQ and MO.

### Table 1. Metrics

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Abbreviation</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Number of Clone Lines</td>
<td>NL</td>
<td>The bigger the NL, the higher the maintenance costs, the more unsuitable for maintenance.</td>
</tr>
<tr>
<td>2</td>
<td>Clone Changing Frequency</td>
<td>CCF</td>
<td>The bigger the CCF, the higher the probability of the occurrence of the inconsistent change, the higher the possibility of maintenance.</td>
</tr>
<tr>
<td>3</td>
<td>Days since Last Clone Change</td>
<td>DLCC</td>
<td>The same as above.</td>
</tr>
<tr>
<td>4</td>
<td>Clone Age</td>
<td>CA</td>
<td>The bigger the CA, the more stable, the recommended package to become the new system construction.</td>
</tr>
<tr>
<td>5</td>
<td>Clone Complexity</td>
<td>CC</td>
<td>The greater the CC, the higher the cost of maintenance.</td>
</tr>
<tr>
<td>6</td>
<td>Maintenance Overhead for Code Clone</td>
<td>MO</td>
<td>Calculated from the aspects 1 to 5</td>
</tr>
<tr>
<td>7</td>
<td>Message Level1</td>
<td>Level1</td>
<td>The level distinguished by Common Weakness Enumeration (CWE). The higher the probability of occurrence, the greater the probability of causing bugs.</td>
</tr>
<tr>
<td>8</td>
<td>Message Level2</td>
<td>Level2</td>
<td>The same as above.</td>
</tr>
<tr>
<td>9</td>
<td>Message Level3</td>
<td>Level3</td>
<td>The same as above.</td>
</tr>
<tr>
<td>10</td>
<td>Message Level4</td>
<td>Level4</td>
<td>The same as above.</td>
</tr>
<tr>
<td>11</td>
<td>Message Level5</td>
<td>Level5</td>
<td>The same as above.</td>
</tr>
<tr>
<td>12</td>
<td>Count of Violations of a Clone</td>
<td>CoV</td>
<td>The higher the CoV value, the worse the quality.</td>
</tr>
<tr>
<td>13</td>
<td>Lack of Software Quality of the Clone</td>
<td>LSQ</td>
<td>Calculated from the aspects 7 to 12</td>
</tr>
<tr>
<td>14</td>
<td>Reconstruction suggestion</td>
<td>Refac</td>
<td>Recognized by MO, LSQ and influenced by the other 13 measures.</td>
</tr>
</tbody>
</table>

#### 3.1. Train the Prediction Model

In this research, we used Bayesian network machine learning method to train the code quality prediction model. All of the discrete metrics of code clone quality above were as training sample data for machine learning method. And each metric corresponded to a sample space. The steps of training prediction model is described as follow:
Step 1: All of the evaluation metrics was discretized into four states of Low, Mid, High and very High. We choose Chi Merge algorithm [26] to discretize the metrics based on our previous research experience. The algorithm is a well-known method for the discretization of supervised.

Step 2: Build the node for Bayesian network and each node corresponded to a clone code quality metric. We used weka tool [27] to build Bayesian network and trained the prediction model, which was able to predict the clone quality when developed the software of new version. For instance we achieved predict the quality by class Id of clone that maybe cloned in new the version.

3.2. Prediction of Code Clone

The prediction process is divided into two cases: (1) Predict the probability of occurrence of a node in Bayesian network in the case of no node information, (2) Predict the probability of occurrence of a result node in the network in the case of ClassId node value information. For instance, the values of LSQ, MO, Refac and all the other nodes can be calculated after the values of ClassId was given.

3.3. Evaluate the Prediction Model

There have been many researches on performance of Bayesian network at home and abroad. Such as the leaning method based on score [28]. The method mainly uses a score criterion to measure the network model and the uniform extent of the data set. And find out the network model, the maximum of the posterior probability of the network, of the highest score. K2 algorithm [29], the Bayesian score function [28] and mountain climbing algorithm are used to optimize the network model, proposed by Cooper and Herskovits is one of the most famous researches. The score function called K2 is a kind of simplification of Bayesian score. In this paper Bayesian network is evaluated by K2 and classifier helped decide the accuracy, robustness, scalability and its like of the network. The output value of the classifier is used as the evaluation index shown in Table 2.

<table>
<thead>
<tr>
<th>Table 2. The Evaluation Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>CCI</td>
</tr>
<tr>
<td>Kappa value</td>
</tr>
<tr>
<td>Coverage</td>
</tr>
<tr>
<td>TP</td>
</tr>
<tr>
<td>FP</td>
</tr>
<tr>
<td>FN</td>
</tr>
<tr>
<td>Precision</td>
</tr>
<tr>
<td>Recall</td>
</tr>
<tr>
<td>F</td>
</tr>
<tr>
<td>CLASS</td>
</tr>
</tbody>
</table>

4. The Experimental Results and Analysis

Experimental environment: Computer processor i3- M350 CPU@ 2.27GHz ×4, memory 4G, OS Ubuntu 12.04 and windows 7. In this paper 71 released versions of 3 open source software called Xorriso, Smalltalk, Bison were used as experimental data, shown in Table 3.
Table 3. Experimental Data

<table>
<thead>
<tr>
<th>Name</th>
<th>Average size</th>
<th>Version number</th>
<th>Average clone class number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xorriso</td>
<td>8.08MB</td>
<td>29</td>
<td>61</td>
</tr>
<tr>
<td>Smalltalk</td>
<td>20.10MB</td>
<td>18</td>
<td>61</td>
</tr>
<tr>
<td>Bison</td>
<td>14.12BM</td>
<td>24</td>
<td>31</td>
</tr>
</tbody>
</table>

Training all of the discrete measures from Bison by Bayesian network to build the prediction model. When the values of ClassId was given we could predict the values of important LSQ, MO, Refac and the other metrics. As shown in figure 5: a clone class was 1 we predicted the probability of LSQ for High was 0.1147, the probability of MO for Low was 0.3523 and the probability of Refac for Y was 0.0953. It means the clone need not to be restructured. While a clone class was 30 the model shown that the probability of LSQ for High was 0.7531, MO for Low was 0.0698 and Refac for Y is 0.7535. The result indicated the clone should been considered more when restructured the clone.

![Figure 1. The Prediction Model](image)

Using K2 algorithm to evaluate the performance of Bayesian network show in table 4, the Refac node, which was the result node, classification evaluation index results show in Table 5.

Table 4. Evaluation Index of Bayesian Network

<table>
<thead>
<tr>
<th>Name</th>
<th>CCI</th>
<th>Kappa</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xorriso</td>
<td>97.28%</td>
<td>0.8877</td>
<td>99.6%</td>
</tr>
<tr>
<td>Smalltalk</td>
<td>99.78%</td>
<td>0.9943</td>
<td>100%</td>
</tr>
<tr>
<td>Bison</td>
<td>98.86%</td>
<td>0.9704</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 5. Evaluation Index of Refac

<table>
<thead>
<tr>
<th>Name</th>
<th>TP</th>
<th>FP</th>
<th>Precision</th>
<th>Recall</th>
<th>F</th>
<th>CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xorriso</td>
<td>0.882</td>
<td>0.012</td>
<td>0.926</td>
<td>0.882</td>
<td>0.904</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>0.998</td>
<td>0.118</td>
<td>0.980</td>
<td>0.988</td>
<td>0.984</td>
<td>N</td>
</tr>
<tr>
<td>Smalltalk</td>
<td>0.992</td>
<td>0.000</td>
<td>1.000</td>
<td>0.992</td>
<td>0.996</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>1.000</td>
<td>0.008</td>
<td>0.997</td>
<td>1.000</td>
<td>0.999</td>
<td>N</td>
</tr>
<tr>
<td>Bison</td>
<td>0.995</td>
<td>0.014</td>
<td>0.962</td>
<td>0.995</td>
<td>0.978</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>0.986</td>
<td>0.005</td>
<td>0.998</td>
<td>0.986</td>
<td>0.992</td>
<td>N</td>
</tr>
</tbody>
</table>

As shown in Table 3, The Kappa value of prediction model was more than 0.8877, indicated the classifier was almost entirely different from the random classification. The value of CCI was at least 97.28% indicated the 14 nodes in the quality prediction model of clonal group were classified correctly. The value of Coverage was close to 100% meant the 14 nodes are almost covered in the classification process. And in table 4 the precision,
recall and F value of Refac for Y were between 0.88 and 1.00 and for N were more than 0.980. So the quality prediction model of clone was able to predict the quality of clones effectively.

5. Summary

In this paper, both the method of quality evaluation of software code and the learning machine method of Bayesian network are applied to the field of clone code. Not only predict the quality of the software unpublished version, but also propose the refactory suggestion. During the experimental process predict the quality of clones from LSQ and MO these two factors. It is a positive method to prevent the propagation of harmful clones, reduce the cost of software maintenance and improve the efficiency of software development in the development of the new version of the software with the prediction results. While this study had the following deficiencies: (1) The process of prediction, some quality tools were used, can not achieve automatically. And the different quality assessment tools would make different quality assessment results. (2) As Bayesian network node used the discrete variable. We had to set the threshold range, would influence the prediction result, and according to the experience for each node to be discretized before trained the model.

References


Authors

**Dongrui Liu**, He was born in 1989, master, student at Inner Mongolia normal university. His current research interests include software engineering and code analysis.

![Dongrui Liu](image1.jpg)

**Dongsheng Liu**, He was born in 1956, master, professor at Inner Mongolia normal university. His current research interests include software engineering and code analysis.

![Dongsheng Liu](image2.jpg)

**Liping Zhang**, She was born in 1974, master, professor at Inner Mongolia normal university. Her current research interests include software engineering and code analysis.

![Liping Zhang](image3.jpg)

**Min Hou**, She was born in 1973, master, lecturer at Inner Mongolia normal university. Her current research interests include software engineering and code analysis.

![Min Hou](image4.jpg)
Chunhui Wang, She born in 1979, master, lecturer at Inner Mongolia normal university. Her current research interests include software engineering and code analysis.