An empirical approach for complexity reduction and fault prediction for software quality attribute

Rajkumar and Viji
Department of Computer Science and Engineering,
SVS College of Engineering,
Coimbatore, Tamil Nadu, India
Email: rajkumarptd28@gmail.com
Email: eceviji_06@yahoo.co.in
*Corresponding author

S. Duraisamy
Department of Computer Applications,
Sri Krishna College of Engineering and Technology,
Coimbatore, India
Email: sdsamy.s@gmail.com

Abstract: Designing the high-quality software is a difficult one due to the high complexity and fault prone class. To reduce the complexity and predict the fault-prone class in the object orient software design proposed a new empirical approach. This proposed approach concentrates more on to increase the software quality in the object oriented programming structures. This technique will collect the dataset and metric values from CK-based metrics. And then complexity will be calculated based on the weighted approach. The fault prediction will be done, based on the low usage of the dataset and high complexity dataset. This helps to increase the software quality. In simulation section, the proposed approach has performed and analysed the parameters such as accuracy, fairness, recall, prediction rate and efficiency. The experimental results have shown that the proposed approach increases the prediction rate, accuracy and efficiency.

Keywords: complexity reduction; fault prediction; software design; software quality; CK-based metrics.

Reference to this paper should be made as follows: Rajkumar, Viji and Duraisamy, S. (2018) ‘An empirical approach for complexity reduction and fault prediction for software quality attribute’, Int. J. Business Intelligence and Data Mining, Vol. 13, Nos. 1/2/3, pp.177–187.

Biographical notes: Rajkumar is currently working as an Assistant Professor at the Department of Computer Science and Engineering in SVS College of Engineering, Coimbatore, Tamil Nadu, India. He has more than seven years of teaching and industrial experience. His research interests include software engineering, web service; open source tools and computer networking.

Viji is currently working as an Assistant Professor at the Department of Computer Science and Engineering in SVS College of Engineering, Coimbatore, Tamil Nadu, India. She has more than five years of teaching experience. Her research interests include software engineering, mobile computing, cryptography and computer networking.
1 Introduction

Designing of a software system is more complex, integrated and difficult task occur due to the variety of technologies. These technologies are more concentrated on producing the high quality of dataset. By generating the quality of dataset is based on the architecture designing and not at the code designing process. To design the perfect architecture for the high quality of the product. Software architecture has been generated in the engineering domains. This will generate a software as well as a blueprint for the quality product. Based on this blueprint, process of software will proceed. This product may produce the fully automatic process based on the blueprints and software and this will be applied in the computer systems. This object-oriented methodology is used for the developing the complex system in the objects, classes and models. This object modelling technique included three complementary models such as function model, dynamic model and object model. These three models are called as real world model and these models are adding the implementation process and then that model will be implemented. Generally, this object modelling technique has the set of phases and diagramming techniques. These phases are named as process, and this process of object designing technique has described in the following steps:

- this will generate a real world model by including three different models
- generating the system design and object design
- system design will determine the software architecture
- object design will be based on algorithms and data structures
- these design will be converted into a programming language in the implementation process.

These processes are used to estimate the design of the object-oriented programming structure. Initially, develop the model and then design the system and object. And that will be given to the implementation process and that will produce the programming language. But these approaches may produce the high complexity. These real world model requires the experience of the real world, interviews from the client and observe the knowledge from the domain. These helps to increase the quality as well as satisfying the customer requirement. Complexity is the major issue in the designing if object oriented system design. That will be reduced by using the reconsideration process.
2 Literature survey

Karasneh et al. (2016) suggests the detection and specification of anti-patterns scheme. This anti-pattern generated during the implementation of software and that will be avoided by using the quality of source code. This scheme helps to avoid the fault-prone classes in software development system. Reena and Selvi (2016) explain the software prediction technique and it involves genetic algorithm and bagging technique. Predicting the software defect in software engineering field is a difficult task due to noisy disturbances in the datasets. Devine et al. (2016) presents the cross product reuse to maintain the software quality in the object-oriented systems. Here considered the two processes such as reuse across products and reuse across recovery. This cross product reuse technique does not affect the quality as well as it increases the performance. Morales et al. (2016) discusses the anti-patterns corrections scheme is to avoid the issues occurring during the design of anti-patterns. Here introduced the tree multi-objective metaheuristic approaches for the designer performance. MOcell outperforms the better performance in this anti-patterns correction scheme. Yazdi et al. (2016) suggest the model driven software development scheme with the help of three different models such as ARMA, GARCH and the combination of ARMA and GARCH model. These models help to increase the performance and reduce the fault-prone classes.

Martin et al. (2016) presents the analysis of app store to present the technical and non-technical information. This technique helps to design the software designing, resource planning, testing, security and engineering fields. Compared to other approaches, this research work helps to increase the quality and avoid the fault prediction model. Jabangwe et al. (2015) explains the measuring parameters for the object oriented designs. This technique involves the empirical evaluation and it is used to increase the quality of the source code. This measuring parameter is used to measure the flexibility, reliability, effectiveness, maintainability and functionality. This will concentrate more on the quality awareness of the object-oriented designs. Nicolaescu et al. (2015) discusses the issues of the object-oriented coupling in the software development system. And this helps to avoid the gap between the current metric and increase the quality metrics. Chong and Lee (2015) present the weighted complex network for the object-oriented software development system. Based on this structural characteristic, this will analyse the maintainability and reliability of the object-oriented software development system. Suresh (2015) suggests the software development life cycle is used to predict the fault tolerance and reduce the cost. Apache integration framework applied in the software development life cycle, and that will predict the classier models. This helps to increase the efficiency and maintain the correctness in the software.

3 Research issues

Search-based approach (SBA) (Alexandru et al., 2015) is used to detect the fault-prone classes by using the rank-based smart learning approaches. By using the genetic algorithm, tailoring prediction models are used to predict the fault-prone classes in the software engineering fields. With the help of regression tree model and a linear regression model, this scheme helps to predict the defect in software objects. Effort aware
fault-prone (EAFP) prediction model (Ma et al., 2016) for the source code network by using the rank method. This makes the depth exploration for the source code network predict the fault-prone classes. This network measures reduces the values, increases the quality of software designing technique. Coupling and cohesion metrics (CCM) (Kaur and Singh, 2015) is used to predict the fault tolerance in object-oriented programming structure. By using the regression model, this object-oriented design helps to find out the faults in the architecture model. Chidambae and Kemmerer’s metric are suitable for this approach and this has designed in this approach for the fault prediction analysis. Software defect prediction (SDP) (He et al., 2015) is a technique is used to find the software defect in the simplified metric software. Here used the three different types of metrics for the SDP in the software metric. That three different metrics are Top k metrics, redundant metrics and minimum metric. This will build these metric based on their respective scenario. That helps to predict the fault with limited source available. CK metric threshold (CKMT) (Kaur and Singh, 2015) is another method to predict the fault using the log transformation. The log transformation process is used to increase the quality metric in the software field. After the log transformation process, assign the threshold rate. Based on that threshold rate, fault prediction will be done. Compared to the without transformation process, with log transformation process has predicted the fault genuinely.

### 4 Methodology

Quality inspection is based on the complexity of the software designing. The complexity of the software can be determined by using the designing phases such as system design and object design. The objective of this proposed work is to predict the fault-prone classes and complexity level during the designing phase. Generally, object-oriented system design has three steps to generate the programming language. The first step is assigning the model, the second one is designing the phase and the third one is implementation. During the second step, designing phase generates the complexity and fault-prone classes. That can be reduced by using the proposed methodology. By using the k-nearest neighbour model can predict the fault-prone classes. During the software testing process and receiver, the investigation can reduce the complexity and also can predict the fault-prone classes. To increase the software quality and efficiency of this object oriented programming structure, can make the software as a reusable process.

### 5 Proposed work

This proposed work is mainly focused on to complexity reduction and predict the fault-prone class in the object-oriented system design. The steps involved in this proposed approach are data collection, metric selection, and complexity calculation and fault prediction. Figure 1 shows that the process of proposed approach. The data processing will select the software module. Software metric is used to select the metric values by using the CK metrics. Complexity calculation is based on the weighted approach. Fault prediction will allow to predict the fault rate and increase the accuracy rate.
Data will be collected from using the JEdit Java programming language. This will be able to access the 130 programming languages. The different number of classes and datasets are collected from JEdit with different versions. JEdit is the programming text editor and this was developed by using the Java programming language. This JEdit text editor includes 274 classes and this proposed approach considered five classes for the complexity calculation.

5.1 Metric selection

There is a different type of metrics are used to predict the fault in the software engineering field. This proposed work chooses the five different types of metrics, for fault prediction scheme. That selected five metrics are described in Table 1:

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBO</td>
<td>Coupling between objects is used to count the number of classes to be coupled.</td>
</tr>
<tr>
<td>LOC</td>
<td>Lack of cohesion is used to calculate the dissimilarity in the classes</td>
</tr>
<tr>
<td>NOC</td>
<td>Number of children will count the number of subclass in hierarchy class</td>
</tr>
<tr>
<td>DOI</td>
<td>The depth of inheritance is used to calculate the number of steps from the root node to child node in the hierarchy class.</td>
</tr>
<tr>
<td>NPM</td>
<td>A number of public methods are used to calculate the number of public methods in the class.</td>
</tr>
</tbody>
</table>

These are the metrics used in the proposed approach, and these metric values are collected from CK metrics.
5.2 Complexity calculation

The complexity metric can be calculated by using weighted class metric (WCM). This will be suitable for object-oriented programming structure. During the designing process, it produces the high complexity. Complexity can be measured based on the number of classes and number of metrics involved in the designing programming structure.

Definition: Consider the class as $S$ and the metrics as $T_1$ …… $T_k$ are defined. Let $S_1, S_2$ …… $S_k$ are the complexity of the metrics. Then,

\[ WCM = \sum_{j=1}^{k} S_j \]  

If the complexity of the metrics are considered as unity, then,

\[ WCM = k \]  

where $k$ defines the number of metrics.

If the complexity of the metrics is not considered as unity, it will use the McCabe’s cyclomatic complexity (MCC). Then WCM will be generated in equation (3).

\[ WCM = \sum_{j=1}^{k} (MCC)_j \]  

This weighted calculation is suitable for the dynamic complexity metric. For the static complexity metric, generated the average complexity metric and that will be defined in equation (4).

\[ ACM = \left( \frac{1}{k} \right) \sum_{j=1}^{k} S_j \]  

Table 2

<table>
<thead>
<tr>
<th>Classes</th>
<th>Class number</th>
<th>Metric values</th>
<th>WCM</th>
<th>ACM</th>
</tr>
</thead>
<tbody>
<tr>
<td>wxObject</td>
<td>1</td>
<td>1.5</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>wxMenu</td>
<td>2</td>
<td>1.8</td>
<td>6</td>
<td>1.5</td>
</tr>
<tr>
<td>wxTimer</td>
<td>3</td>
<td>2.3</td>
<td>4</td>
<td>0.7</td>
</tr>
<tr>
<td>wxWindow</td>
<td>4</td>
<td>2.6</td>
<td>6</td>
<td>0.8</td>
</tr>
<tr>
<td>wxEvent</td>
<td>5</td>
<td>2.9</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

Here observing the five classes such as wxObject, wxMenu, wxTimer, wxWindow and wxEvent. Table 2 shows that the complexity metrics values for these five classes. For this different class, complexity will be reduced. If any of the class has a defect and that will be defined as a faulty class. If any of the class has no defect, and that will be defined as a non-faulty class. Fault prediction is another important task in this proposed approach.

The complexity will be determined by using the below equation;
An empirical approach for complexity reduction and fault prediction

\[ Ce = \{H, L\} \begin{cases} \{H \in \text{High complexity} \\ \{L \in \text{Low Complexity} \}\end{cases} \]  \hspace{1cm} (5)

Based on the equation (5), complexity will be determined.

5.3 Fault prediction

This fault prediction is calculated based on the complexity level and usage level. The high failure rate is directly proportional to the high fault-prone class.

The usage level may vary gradually based on the user requirement and that will be defined in equation (6).

\[ Uu = \{H, M, L\} \begin{cases} \{H \in \text{High Usage} \\ \{M \in \text{Medium Usage} \\ \{L \in \text{Low Usage} \}\end{cases} \]  \hspace{1cm} (6)

The failure level is the combination of the usage and complexity level occurred during the design of software. The lesser usage level tends to increase the failure rate and that will be considered as fault classes.

The failure level is defined in equation (7).

\[ Ff = Ce \ast Uu \]  \hspace{1cm} (7)

\[ Ff = \begin{cases} Uu \cup \{L - \text{Low Usage}\} \\ Ce \cup \{H - \text{High complexity}\} \end{cases} \]  \hspace{1cm} (8)

The combination of low usage and high complexity is defined as the fault-prone classes in the object-oriented software design. That will be calculated by using equation (8).

6 Results and discussions

This proposed approach has performed for the complexity reduction and fault prediction for the software quality attribute. The software quality has analysed based on the parameters such as accuracy, precision rate, fairness, prediction rate, recall and efficiency.

Accuracy is defined as the ratio of a summation of true positive and true negative to the summation of positive and negative.

\[ \text{Accuracy} = \frac{Tp + Fp}{P + N} \]  \hspace{1cm} where \( P = Tp + Fn \) and \( N = Fp + Tn \).

The recall is defined as the ratio of true positive to the summation of true positive and false negative.

Precision is defined as the ratio of true positive to the summation of true positive and false positive.
Table 3  Parameter description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_p$</td>
<td>True positive Faulty class and faulty metric</td>
</tr>
<tr>
<td>$F_p$</td>
<td>False positive Faulty class and non-faulty metric</td>
</tr>
<tr>
<td>$T_n$</td>
<td>True negative Non-faulty class and non-faulty metric</td>
</tr>
<tr>
<td>$F_n$</td>
<td>False negative Non-faulty class and faulty metric</td>
</tr>
</tbody>
</table>

Fairness is defined as the square of the weighted precision and recalls to the summation of precision and recall with square weighted magnitude.

$$F_{ns} = \frac{(W^2 + 1) \cdot Pre \cdot Rec}{W^2 \cdot Pre + Rec}$$

where $w$ is weight, if $w = 1$, precision and fairness have equal weights.

Prediction rate is calculated based on the fault prediction based on the combination of usage level and complexity level.

Efficiency is also known as software quality, this can be calculated based on the overall performance.

The proposed approach has compared with the existing approaches such as SBA, EAFP prediction model and SDP approach. By varying the different classes, corresponding metric values are collected from CK metrics. By varying the metric classes, the performance of the existing and proposed approach has analysed.

Figure 2  Accuracy rate

Figure 2 shows that the accuracy rate by varying the metric values with different classes. The proposed approach achieves 96%. The existing approaches SBA, EAFP prediction model and SDP approach have lesser accuracy rate due to the lesser prediction model. The proposed approach increases the accuracy rate with the help of complexity reduction and fault prone class prediction in object-oriented software design.

Figure 3 show that the recall as well as sensitivity, the proposed approach increases the sensitivity rate versus metric values. The existing approaches are SBA, EAFP.
prediction model and SDP approach has a lesser recall. But the proposed approach increases the sensitivity of 90% with the help of complexity reduction and the high quality of software.

**Figure 3** Analysis of recall

![Figure 3](image)

**Figure 4** Analysis of fairness

![Figure 4](image)

Figure 4 shows that the fairness rate versus metric values. The existing techniques such as SBA, EAFP prediction model and SDP approach have lesser fairness due to the high complexity. But, the proposed approach increases the fairness rate by reducing the complexity.

Figure 5 shows that the prediction rate. By detecting the fault correctly is known as prediction rate. The proposed approach has high prediction rate and which based on the low usage level and high complexity level. The proposed approach has 95% accurate prediction rate. Compared to the existing SBA, EAFP prediction model and SDP approaches, proposed approach has high prediction rate.
Figure 6 shows that the efficiency analysis. Compared to the existing approaches SBA, EAFP prediction model and SDP approach, the proposed approach has high efficiency due to the low complexity and accurate prediction rate. This proposed approach helps to increase the software quality attribute in the object-oriented software design technology.

**Figure 5** Prediction rate

![Prediction rate chart]

**Figure 6** Efficiency analysis

![Efficiency analysis chart]

7 Conclusions

The proposed approach is used to achieve the software quality attribute in the object-oriented system design by reducing the complexity as well as predicting the fault-prone classes. To reduce the complexity uses the weight-based complexity
calculation. To predict the fault-prone classes uses the low usage level of datasets and high usage level of complexity. By reducing the complexity and predicting the fault, this proposed approach helps to increase the software quality during the system design. The experimental results analyse the performance of the proposed approach and compared with the existing approaches. The proposed approach increases the accuracy, fairness, recall, prediction rate and efficiency compared to SBA, EAFP prediction model and SDP approaches.

References