Analysis of Structural Coupling Strength in Improving Software Module Structure

Amarjeeta a, Jitender Kumar Chhabra a

a Department of Computer Engineering, NIT Kurukshetra, Haryana, India, Contact: amarjeetnitkk@gmail.com

The organization of a software system widely affects the various quality parameters such as modifiability, understandability, and testability. Hence, the success of medium or large scale software systems heavily depends on how well they are organized. However, systems often require to be modified to satisfy the environment and requirements. Mostly, it is carried out without following the original design principles of the system. Over a period of time, such a continuous modification deteriorates the system organization, hence increases the system complexity. To improve the system structure, the software clustering seems more realistic technique. Recently, the search-based software engineering approach gain more attention to solve the software clustering problem. In this paper, we propose a search-based multi-objective optimization to improve the structure of the object-oriented software systems. To determine the coupling strength between the software artifacts, we uses different coupling strength scheme such as binary coupling, additive coupling and relative coupling scheme. The approach is evaluated over four real-world and three random software applications. The experimentation results show that how the use of additive and relative coupling strength scheme leads to generate more effective solutions compared binary coupling strength.

Keywords: Modularization Quality, Multi-objective Optimization, Search based Software Engineering, Software Restructuring.

1. INTRODUCTION

A medium or large scale software system often needs constant modification that is triggered due to evolving technologies, changing requirements and stakeholder [1]. It has been revealed that the maintenance of software system, cost up to 75% of total software development cost [2]. In order to fix the bugs or satisfy the new requirements, maintainers modify the software system. Generally, the software developers do not follow the guidelines given in the software documentation during the modification, which leads system structure erosion [3]. There can be many reasons not to follow the documentation guidelines, for instance restrict delivery, lack of proper developer training, and absence of long-term commitment developers to the project [4]. Such maintenance practices deteriorate the system structure and leads system more complex for future maintenance [5]. Hence, to make system more modifiable it requires improving the system structure. To improve the system structure, software refactoring plays an important role, where the system structure is modified without changing the external behavior [6]. Applying software refactoring to re-structure the whole system is often challenging and costly [7]. Software clustering is a most feasible technique that can be used to obtain refactoring of whole system more efficiently. Hence, software clustering is widely used as an activity of software refactoring for whole system. Software clustering techniques has been an active research for more than twenty years. In literature, the software clustering has been performed at various levels of software granularity, with different clustering approaches [8]. A basic concept often used by the existing approaches is that soft-
maximum improvement is 38.6%. If we compare the three coupling strength scheme, the additive coupling strength scheme and relative coupling strength scheme perform better than the binary coupling strength in most of the problem instances. Now, if we compare the additive coupling strength scheme and relative coupling strength scheme, the Table shows that the relative coupling strength scheme perform better than the additive coupling strength scheme with all problem instance.

Table I: Information about the Original Software Applications

<table>
<thead>
<tr>
<th>Systems</th>
<th>Version</th>
<th>#Classes</th>
<th>#Connections</th>
<th>#Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>EroCC</td>
<td>1.5</td>
<td>154</td>
<td>722</td>
<td>6</td>
</tr>
<tr>
<td>JClass</td>
<td>3.81</td>
<td>100</td>
<td>276</td>
<td>6</td>
</tr>
<tr>
<td>Java Servlet API</td>
<td>2.3</td>
<td>65</td>
<td>113</td>
<td>4</td>
</tr>
<tr>
<td>VML API DOM</td>
<td>1.03d</td>
<td>110</td>
<td>209</td>
<td>9</td>
</tr>
<tr>
<td>DOM 47</td>
<td>1.32</td>
<td>105</td>
<td>130</td>
<td>16</td>
</tr>
<tr>
<td>Random50</td>
<td>NA</td>
<td>50</td>
<td>134</td>
<td>7</td>
</tr>
<tr>
<td>Random100</td>
<td>NA</td>
<td>100</td>
<td>295</td>
<td>12</td>
</tr>
<tr>
<td>Random150</td>
<td>NA</td>
<td>150</td>
<td>212</td>
<td>18</td>
</tr>
</tbody>
</table>

Table II: Mean and Standard Deviation of MQ

<table>
<thead>
<tr>
<th>Systems</th>
<th>Mean</th>
<th>SD</th>
<th>Mean</th>
<th>SD</th>
<th>Mean</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>EroCC</td>
<td>2.12</td>
<td>0.11</td>
<td>22.55</td>
<td>3.06</td>
<td>26.23</td>
<td>2.99</td>
</tr>
<tr>
<td>JClass</td>
<td>3.84</td>
<td>0.11</td>
<td>32.24</td>
<td>3.25</td>
<td>16.88</td>
<td>3.54</td>
</tr>
<tr>
<td>Java Servlet API</td>
<td>2.12</td>
<td>0.14</td>
<td>22.25</td>
<td>3.06</td>
<td>24.85</td>
<td>2.97</td>
</tr>
<tr>
<td>VML API DOM</td>
<td>3.84</td>
<td>0.11</td>
<td>29.29</td>
<td>3.55</td>
<td>22.34</td>
<td>2.99</td>
</tr>
<tr>
<td>DOM 47</td>
<td>2.12</td>
<td>0.11</td>
<td>22.54</td>
<td>3.06</td>
<td>22.54</td>
<td>2.99</td>
</tr>
<tr>
<td>Random50</td>
<td>3.84</td>
<td>0.11</td>
<td>32.24</td>
<td>3.06</td>
<td>24.85</td>
<td>2.97</td>
</tr>
<tr>
<td>Random100</td>
<td>2.12</td>
<td>0.11</td>
<td>22.54</td>
<td>3.06</td>
<td>22.54</td>
<td>2.99</td>
</tr>
<tr>
<td>Random150</td>
<td>3.84</td>
<td>0.11</td>
<td>32.24</td>
<td>3.06</td>
<td>24.85</td>
<td>2.97</td>
</tr>
</tbody>
</table>

Figure 2. Percentage Improvements in MQ Values with Various Coupling Scheme

The Figure 2 summarizes the results of binary coupling, additive coupling and relative coupling scheme. Basically, it shows the percentage improvement in MQ values of all three coupling strength scheme. The horizontal axis shows the problem instances and the vertical axis sows the percentage improvement in the MQ values.

8. CONCLUSION AND FUTURE WORKS

In this paper, we proposed a search-based multi-objective optimization technique to improve the modular structure of an object-oriented software system. The approach is evaluated over eight software systems where the class coupling is determined using various coupling strength scheme namely binary coupling, additive coupling and relative coupling strength scheme. The experimentation results show that the approach with each coupling strength scheme, improves the MQ value of each problem instance. However, the proposed approach with additive coupling and relative coupling schemes perform better compared to binary coupling scheme over maximum problem instances. On the other hand, the relative coupling scheme performs better in comparison of additive coupling scheme over all the problem instances. Hence, the results conclude that the performance of the multi-objective optimization approach can be improved with the application of suitable coupling scheme. The
future works includes the formulation and evaluation of more coupling schemes with multi-objective optimization techniques for improving the object-oriented software structures.

REFERENCES


32. http://www.stan4j.com