A Complexity Measure Based Object-Oriented Software Metrics
1 Elnaz Namazi, 2 Mahdi Bazargani

1 Department of Information Technology Engineering, Zanjan Branch, Islamic Azad University, Zanjan, Iran
2 Department of Computer Engineering, Zanjan Branch, Islamic Azad University, Zanjan, Iran

1 raznep@gmail.com, 2 mbzir@gmail.com

ABSTRACT
A key task in software engineering is the ability to estimate the cost and time to deliver. There are many characteristics of software projects that can be measured, one of the parameters affecting those, is the complexity of it. The relationships between complexity and coupling of object oriented software have been studied for the past several years. The aim of this paper is to calculate the complexity, in order to estimate the costs of the software before we start coding. We proposed a model for estimate complexity using methods, variables and coupling between class diagrams. This complexity measure was evaluated against class diagrams, developed in Java.

Keywords: Complexity, Object-Oriented, Parallel algorithms, Software Metrics.

1. INTRODUCTION
Estimating the complexity using software metrics is one of the main parameters that were made by researchers in order to avoid complex software [1]. The goal of this metrics is to minimize the development complexity, time, effort and cost by using the reusable software components [2,3]. There are different metrics each focusing on different complexity factors [4]. Among these metrics most commonly used that Line of Code (LOC) [5], McCabe’s Cyclomatic, Halstead’s complexity metrics and Kafura’s and Henry’s fan in-fan-out [6,10]. Nowadays Component Based Software Developments (CBSD) is the newest method for the implementation of large and complex software [3,12]; But those Methods are not appropriate for Object Oriented programming; Complexity of object-oriented program based on the number of classes and modules, encapsulation and inheritance and abstraction information hiding is measured [11], one of the popular methods for designing OO is Chidamber and Kemerer’s method [7,8]; Chidamber and Kemerer suggested new metric by proposed Weighed methods per class, Depth of Inheritance, Response For Class, Coupling Between Objects, Lack of Cohesion Method and Number of Children [6, 9].

2. MATERIALS AND METHODS
The aim of this paper is suggested a model to estimate and predict the coding complexity classes before starting programming.

According to customer requirements trying to achieve these objectives, the project has collected and analyzed in the design stage to draw class diagrams, and on the number and type of methods classes to complexity estimate.

The way it works is that having class diagram, input and output methods identified and the type method in four categories: simple, medium, complex and very complex and assign weights to them.

Theorem 1: We estimate methods complexity with using input arguments and outputs; so for sum of the weight of methods on number of them show the methods complexity.

\[
\text{Complexity of Methods} = \sum \text{Weight of Methods} \times \text{Number of Methods}
\]

Theorem 2: We estimate variable complexity by dividing into four groups, such as simple, medium, complex and high complex. And estimate it as follow:

\[
\text{Complexity of Variables} = \sum \text{Weight of Variables} \times \text{Number of Variables}
\]

Theorem 3: The coupling between methods of class estimate by called methods per declared methods.

\[
\text{Complexity of Line} = \frac{\text{Called Methods}}{\text{Declared Methods}}
\]

Theorem 4: We assume that the complexity of diagrams depends on what contributes to develop diagrams. (So in an Object Oriented (OO) paradigm, diagrams may consist on classes and dependency between them and complexity of parallel algorithms. The complexity of classes can be estimated by complexity of methods, variables and cohesion and coupling in classes.

This may be defined as shown in the following equation:
3. DISCUSSIONS

The proposed method is tested and to get the values of the above metrics, an experiment is conducted on some open source JavaBeans class diagrams. Methods due to the complexity of the diagram for each class and type of input and output methods and estimates used to verify the readability will be according to the dependency with increasing complexity, readability decreases and vice versa. The Realationship between readability and complexity is showing in figure 1.

![Fig 1: Realationship between readability and complexity](image)

And results of the calculated complexity are given in Table 1. Where CD stands for Class Diagram, NC stands for Number of Classes, NM stands for Number of Methods, NGS stands for Number of Get/Set methods, RD stands for Readability and CC stands for complexity of classes in class diagrams. Realationship between complexity and number of Get/Setting methods.

<table>
<thead>
<tr>
<th>CD</th>
<th>NC</th>
<th>NM</th>
<th>NGS</th>
<th>RD</th>
<th>CC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>12</td>
<td>5</td>
<td>0.871</td>
<td>2.80</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>17</td>
<td>6</td>
<td>0.895</td>
<td>1.50</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>26</td>
<td>8</td>
<td>0.610</td>
<td>6.8</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>28</td>
<td>14</td>
<td>0.48</td>
<td>6.35</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>27</td>
<td>10</td>
<td>0.52</td>
<td>6.55</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>35</td>
<td>21</td>
<td>0.33</td>
<td>9.85</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>30</td>
<td>15</td>
<td>0.75</td>
<td>3.95</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>33</td>
<td>18</td>
<td>0.421</td>
<td>8.87</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>30</td>
<td>15</td>
<td>0.498</td>
<td>7.94</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>35</td>
<td>21</td>
<td>0.403</td>
<td>8.65</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>42</td>
<td>24</td>
<td>0.398</td>
<td>9.54</td>
</tr>
<tr>
<td>12</td>
<td>25</td>
<td>56</td>
<td>31</td>
<td>0.325</td>
<td>10.31</td>
</tr>
</tbody>
</table>

Complexity Performance shows in figure 2, such as we prospect with increase complexity, decread readability and increse number of Get/Set methods.

![Fig 2: Realationship between number of Get/set methods and complexity](image)

Also we compare our supose with last methods and it shows in figure 3. In this chart the light bar is showing the our method and dark bar is showing the last method.

![Fig 3: Performance compare with last methods](image)

In this paper, we proposed a model for estimate complexity using methods, variables and coupling between class diagrams. The result shows that there exists inversely proportional relation between the rate of complexity and readability. When we interpret it means that low complexity lead to the high readability thus results in low implementation time.
4. CONCLUSION

For measuring and estimating software cost, complexity metrics have effect on it which need to be identified before the development takes place. Considering the paradigm of object oriented programming, one of the issues is to estimate the complexity of it and what parameters are being affected.

In this study, we aimed to suggest a method in that estimates the complexity of classes based on the complexity of the classes, methods and variables, the data types of inputs and outputs, and dependencies between classes that caused by call methods and data, so we have studied class diagram and compared the quality of it with readability of parameter.

REFERENCES


