CHAPTER 9

Unified Measure of OO Complexity

9.1 Introduction

Complexity has been used to characterize software. Banker et al. state, "Software complexity refers to the extent to which a system is difficult to comprehend, modify and test, not to the complexity of the task which the system is meant to perform; two systems equivalent in functionality can differ greatly in their software complexity" [99]. Software complexity is defined in IEEE Standard 729-1983 as: "The degree of complication of a system or system component, determined by factors such as the number and intricacy of interfaces, the number and intricacy of conditional branches, the degree of nesting, the types of data structures, and other system characteristics." According to McCall, a complexity relates to data set relationships, data structures, data flows and data being implemented and measures the degree of decision – making logic within the system [100].

9.2 Complexity Measurement of Object Oriented Systems

Object Oriented software development requires a different approach from traditional development methods, including metrics used to evaluate the software [101-107]. Chidamber and Kemmerer [108] have proposed the following 6 metrics - WMC, RFC, LCOM, CBO, DIT and NOC, for measuring the OO software quality attributes. The following OO software quality attributes can be measured:

- Efficiency -- are the constructs efficiently designed?
• Complexity -- could the constructs be used more effectively to decrease the architectural complexity?

• Understandability -- does the design increase the psychological complexity?

• Reusability -- does the design quality support possible reuse?

• Testability and maintainability -- does the structure support ease of testing and changes?

It is important to mention that depending on the size and complexities of the project, the set of metrics needs to be adjusted to get the right benefit. The decision of which ones to use is mostly down to the organizational practice and experience of individual professionals. Having selected the subset of metrics the other key issue faced today is that there is no unified measure of OO software complexity.

For the purpose of this study we are considering a subset of the metrics suite, provided by Chidamber and Kemerer [108], for object oriented design measurement that are applied by the Software Assurance Technology Center (SATC) at NASA Goddard Space Flight Center [98]. These metrics have been used in the evaluation of many NASA projects and empirically supported guidelines have been developed for their interpretation. The metrics used in this model are WMC, CBO, RFC, and DIT. These four metrics are selected as these metrics have the impact on the complexity. The threshold values for the individual metrics need to be derived by discussing with project managers and programmers and studying the distributions of the metrics collected over a period of time. The methodology is to combine OO software metric values into a single overall value that can be used to rank classes according to their complexity using fuzzy techniques and concepts.
9.3 Proposed Fuzzy Model for Object Oriented Complexity

A major problem that exists with this approach is that it is very difficult to quantify the different attributes of software and their values are usually approximated to some extent. To overcome this problem we propose a fuzzy model for complexity measurement because fuzzy model is the best thing to represent such doubtful, ambiguous and diverging opinions. Fuzzy logic variables are used for defining the values for different metrics, which will serve as the inputs to our fuzzy model. This model takes into consideration the effects of WMC, CBO, DIT and RFC on the complexity of software. It consists of 4 modules. The fuzzification module is the first step in working of any fuzzy model, which transforms the crisp input(s) into fuzzy values. In the next stage, these values are processed in fuzzy domain by inference engine based, on the knowledge base (rule base or production rules) supplied by domain expert(s). During this stage, the fuzzy operators are applied, the implication process is followed, and then all outputs are aggregated. Finally the processed output is transformed from fuzzy domain to crisp domain by defuzzification module.

9.3.1 Membership Functions of Variables

First of all the inputs WMC, RFC, CBO and DIT are fuzzified into the membership functions as shown in figures given below. The level values are just the indicative figures provided by project managers and programmers of NASA. The threshold values need to be derived by collecting and analyzing data over a period of time.

Input parameters for the Proposed Model

WMC is a weighted sum of the methods in a class [109]. The Cyclomatic Complexity [79,110] is used to evaluate the minimum number of test cases needed for each method. The
number of methods and the complexity of those methods are a predictor of how much time and effort is required to develop and maintain the class.

**FIGURE 9.1: MEMBERSHIP FUNCTION FOR WMC METRIC**

RFC is the cardinality of the set of all methods that can be invoked in response to a message to an object of the class or by some method in the class [109]. The larger the number of methods that can be invoked from a class through messages, the greater the complexity of the class, complicating testing and debugging due to ripple effect.

**FIGURE 9.2: MEMBERSHIP FUNCTION FOR RFC METRIC**

CBO is a count of the number of other classes to which a class is coupled. It is measured by counting the number of distinct non-inheritance related class hierarchies on which a class depends [109]. Coupled classes must be bundled or modified if they are to be reused. A high CBO indicates classes that may be difficult to understand, reuse or maintain. The larger the CBO, the higher the sensitivity to changes in other parts of the design and therefore
<table>
<thead>
<tr>
<th># Of OO Metrics deviating from threshold values</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;= 1</td>
<td>Simple class, low</td>
</tr>
<tr>
<td>1 – 2</td>
<td>More complex class, medium</td>
</tr>
<tr>
<td>&gt; 2</td>
<td>High Complex class, high</td>
</tr>
</tbody>
</table>

Table 9.1: Term Set for Output Variable Complexity

9.3.3 Rule Base For the proposed Model

In the proposed fuzzy model we are considering four inputs each consisting of three terms therefore our rule base consists of 81 rules after considering all the possible combinations of inputs. Suppose if a fuzzy model is having n inputs each consisting of m terms then the possible number of rules, say R, for this model can be calculated by considering the Cartesian product of all the input states.

\[
R = M \times M \times M \ldots \text{upto n times}
\]  \hspace{1cm} (9.1)

Or \[
R = M^n
\]  \hspace{1cm} (9.2)
Our rule base can be represented in a tabular form consisting of 81 rows as shown in table 9.2.

<table>
<thead>
<tr>
<th>S. No</th>
<th>CBO</th>
<th>WM C</th>
<th>RFC</th>
<th>DIT</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PR</td>
<td>PR</td>
<td>PR</td>
<td>PR</td>
<td>LOW</td>
</tr>
<tr>
<td>2</td>
<td>AC</td>
<td>PR</td>
<td>PR</td>
<td>PR</td>
<td>LOW</td>
</tr>
<tr>
<td>3</td>
<td>NAC</td>
<td>PR</td>
<td>PR</td>
<td>PR</td>
<td>LOW</td>
</tr>
<tr>
<td>80</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>81</td>
<td>NAC</td>
<td>NAC</td>
<td>NAC</td>
<td>NAC</td>
<td>HIGH</td>
</tr>
</tbody>
</table>

Table 9.2: Rule Base of the Fuzzy Model

A pictorial representation of Rule Base consisting of 81 rules is shown in Figure 9.5.

9.4 Working of Fuzzy Model

First of all the input data i.e. crisp values for inputs are fed to the fuzzification module. After that the degree of participation of each input corresponding to their fuzzy set is determined [106]. Then inference engine starts processing of these inputs on the basis of which some rules get fired which gives the fuzzified output.

For selection of rules all the input states are considered simultaneously i.e. these inputs are connected by AND operator therefore we are using min fuzzy operator to find the degree of membership of firing [68,81,82]. If more than 1 rule get fired then to aggregate the effects of all the fired rules max aggregation operator is used and after that defuzzification of the fuzzified output is done. We are using the center of gravity method for this purpose.
9.5 Sample Output Computation For The Model

Suppose we have the following crisp inputs to the model:

\[ \text{WMC} = 10, \text{DIT} = 5, \text{RFC} = 20, \text{CBO} = 1 \]

These inputs are fed to the fuzzification module and after the fuzzification of the given values we find that WMC = 10 belongs to fuzzy set preferred (PR) with membership grade 1, DIT = 5 belongs to the fuzzy set not acceptable (NA) with membership grade 1, RFC = 45 belongs to fuzzy set acceptable (AC) with membership grade of 0.25 and to fuzzy set not acceptable (NA) with membership grade of 0.75 and CBO = 1 belongs to fuzzy set preferred (PR) with the possibility of a degree of 1. With these input values we find that following rules get fired:

<table>
<thead>
<tr>
<th>WMC</th>
<th>RFC (45)</th>
<th>CBO</th>
<th>DIT</th>
<th>COMPLEXITY</th>
<th>Membership grade of complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>(10)</td>
<td>AC</td>
<td>PR</td>
<td>NA</td>
<td>LOW</td>
<td>Min (1,0.25,1,1) = 0.25</td>
</tr>
<tr>
<td>PR</td>
<td>NA</td>
<td>PR</td>
<td>NA</td>
<td>MEDIUM</td>
<td>Min (1,0.75,1,1) = 0.75</td>
</tr>
</tbody>
</table>

Table 9.3: Complexity Calculation for given input
FIGURE 9.5: PICTORIAL REPRESENTATION OF RULE BASE

FIGURE 9.6: OUTPUT VARIABLE COMPLEXITY AGGREGATION
9.6 Defuzzification

After getting the fuzzified outputs as shown above, we defuzzify them to get the crisp value of the output variable complexity [83,84]. For this we are finding the Centre Of Gravity (COG) of the above fuzzy output as follows:

\[
COG = \frac{\int \mu x \, dx}{\int \mu \, dx} \tag{9.3}
\]

\[
COG = \frac{\left\{ \left(\frac{0.25 x}{225} + \frac{y x}{225} + \frac{0.75 x}{325} + \frac{4 y x}{325} \right) \right\} \left\{ \left(\frac{225 y}{0} + \frac{275 y}{0} + \frac{325 y}{0} + \frac{4 y}{0} \right) \right\}}{\left\{ \left(\frac{0.25 x}{225} + \frac{y x}{225} + \frac{0.75 x}{275} + \frac{4 y x}{325} \right) \right\} \left\{ \left(\frac{225 y}{0} + \frac{275 y}{0} + \frac{325 y}{0} + \frac{4 y}{0} \right) \right\}} \tag{9.4}
\]

\[
COG = \frac{\left\{ \left(\frac{0.25 x}{225} + \frac{(m x + c) x}{275} + \frac{0.75 x}{325} + \frac{(m x + c) x}{325} \right) \right\} \left\{ \left(\frac{225 y}{0} + \frac{275 y}{0} + \frac{325 y}{0} + \frac{4 (m x + c) y}{0} \right) \right\}}{\left\{ \left(\frac{0.25 x}{225} + \frac{(m x + c) x}{275} + \frac{0.75 x}{325} + \frac{(m x + c) x}{325} \right) \right\} \left\{ \left(\frac{225 y}{0} + \frac{275 y}{0} + \frac{325 y}{0} + \frac{4 (m x + c) y}{0} \right) \right\}} \tag{9.5}
\]

\[
COG = \frac{\left\{ \left(\frac{0.25 x^2}{225} + \frac{(x - 2) x}{275} + \frac{0.75 x^2}{325} + \frac{(4 - x) x}{325} \right) \right\} \left\{ \left(\frac{225 y}{0} + \frac{275 y}{0} + \frac{325 y}{0} + \frac{4 (4 - x) y}{0} \right) \right\}}{\left\{ \left(\frac{0.25 x^2}{225} + \frac{(x - 2) x}{275} + \frac{0.75 x^2}{325} + \frac{(4 - x) x}{325} \right) \right\} \left\{ \left(\frac{225 y}{0} + \frac{275 y}{0} + \frac{325 y}{0} + \frac{4 (4 - x) y}{0} \right) \right\}} \tag{9.6}
\]

\[
COG = 2.2996456 \tag{9.7}
\]

We observed the effect of these rules by simulating the proposed fuzzy system with the help of MATLAB- Fuzzy Tool Box [85]. For the above given inputs the value of complexity comes out to be 2.3035 which is very near to our calculated value. One sample surface view of the simulated fuzzy model is shown in the figure 9.7.
The integrated approach gives a true picture of the software complexity. This can prove to be a handy tool for managers to rank the Object Oriented classes based on their complexities. Also, the complexity measurement will aid in decisions related to testing, for example, more complicated class demands for thorough testing and should be given priority over the less complicated ones. Since we are considering an integrated approach for indicating the complexity, therefore, the effect of all the parameters can be adjudged simultaneously.

![Surface View of the Simulated Fuzzy Model](image)

**FIGURE 9.7: SURFACE VIEW OF THE SIMULATED FUZZY MODEL**

9.7 Summary

Presented a single unified measure of software complexity of Object-Oriented software. The proposed model measures the complexity of object-oriented systems based on the four important object-oriented metrics—WMC, RFC, CBO and DIT. This model integrates the effects of different parameters of object-oriented systems, which can affect the complexity and is not biased towards one particular aspect of the system. This model can be useful in deciding about the order of complexities of different classes which in further can help in
determining the risk associated with the classes of different complexity level and we can prioritize the classes on the basis of risk associated with them for regression testing.