Fuzzy Based Approach for Predicting Software Maintainability

Vishal Choudhary#1 and Dr. P. K. Bharti#2

Department of MCA
AIMT, Knowledge Park-3 Greater Noida-(India)

Abstract - Software maintenance is a process of modifying existing operational software while leaving its primary functions intact. Software maintenance encompasses a broad range of activates, like error correction, enhancement of capabilities, deletion of obsolete capabilities and optimization. Software maintainability assessment is a major issue these days. Producing software that is easy to maintain may save large costs in industries. The maintenance of existing software can account for 70% of the total efforts put in application development [Pres05]. The value of software can be enhanced by meeting additional requirements, making it easier to use, improving efficiency and employing newer technologies. This paper discusses various issues and challenges, related with the maintainability assessment of software systems. The present work proposes a fuzzy logic based approach for quantification of maintainability of software system based on combined effect of four major aspects of software, i.e. average number of live variables, average life span of variables, average cyclomatic complexity and comment ratio.

Classroom projects are considered to estimate and validate the proposed maintainability model.

Keyword: Cyclomatic complexity, comment ratio, maintainability, triangular fuzzy number

1. METRICS FOR MAINTAINABILITY ASSESSMENT
Researcher have tried to quantify maintainability in different types of measures [12, 13, 14]. Here we consider four major aspects of software to assess maintainability, i.e. average number of live variables, average life span of variables, average cyclomatic complexity and comment ratio.

a) AVERAGE NUMBER OF LIVE VARIABLES (ALV):
a variable is live at a particular statement only if a certain number of statements reference it before or after that statement. The average number of live variables is the sum of the count of live variables divided by the count of executable statements. The higher, the average number of live variables, the more difficult it would be to develop and to maintain the software.

b) THE AVERAGE LIFE SPAN OF VARIABLES (ALS):
The life span of a variable is defined as the number of statements between two successive references of the same variable. The average life span of variable is the Ratio of the sum of life spans to number of variables.

C) COMMENT RATIO (CR):
The comment ratio is defined as

\[
CR = \frac{S + C}{C} \quad (1)
\]

Where

S denotes total line of code
C represents total number of comment lines.
A lower comment ratio means better readability and the better maintainability.

**D) AVERAGE CYCLOMATIC COMPLEXITY (ACC):**
The average cyclomatic complexity is defined as the average cyclomatic complexity of all modules, where cyclomatic complexity is defined as

\[ V = e - n + 2p \]  

Where \( e \) is number of edges in a program flow graph, \( n \) is the number of nodes, and \( p \) is the number of connected Components. Maintainability of software declines with increasing average cyclomatic complexity.

**2. FUZZY SET THEORY FOR MAINTAINABILITY:**

Recently many researchers have proposed some integrated models for maintainability measurement, which leave significant room for further improvements. Here we proposed a methodology to improve the maintainability metrics system based on the fuzzy set theory. ALV, ALS, ACC and CR are classified in four levels of complexity i.e low, medium high and very high and corresponding weights are then assigned for each. The complexity levels and their corresponding weights for ALV, ALS ACC and CR are described in table 1 and 2

<table>
<thead>
<tr>
<th>ALV</th>
<th>Complexity</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-2</td>
<td>Low</td>
<td>W1</td>
</tr>
<tr>
<td>2-5</td>
<td>Medium</td>
<td>W2</td>
</tr>
<tr>
<td>5-8</td>
<td>High</td>
<td>W3</td>
</tr>
<tr>
<td>8 or more</td>
<td>Very High</td>
<td>W4</td>
</tr>
</tbody>
</table>

**TABLE 2: COMPLEXITY AND WEIGHT FOR ALS**

<table>
<thead>
<tr>
<th>ALS</th>
<th>complexity</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-20</td>
<td>Low</td>
<td>W1</td>
</tr>
<tr>
<td>20-150</td>
<td>Medium</td>
<td>W2</td>
</tr>
<tr>
<td>150-400</td>
<td>High</td>
<td>W3</td>
</tr>
<tr>
<td>400 or more</td>
<td>Very high</td>
<td>W4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ACC</th>
<th>complexity</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-5</td>
<td>Low</td>
<td>W1</td>
</tr>
<tr>
<td>5-13</td>
<td>Medium</td>
<td>W2</td>
</tr>
<tr>
<td>13-18</td>
<td>High</td>
<td>W3</td>
</tr>
<tr>
<td>18 or more</td>
<td>Very high</td>
<td>W4</td>
</tr>
</tbody>
</table>

**TABLE 4: COMPLEXITY AND WEIGHT FOR ACC**

<table>
<thead>
<tr>
<th>ALS</th>
<th>complexity</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-4</td>
<td>Low</td>
<td>W1</td>
</tr>
<tr>
<td>4-8</td>
<td>Medium</td>
<td>W2</td>
</tr>
<tr>
<td>8-12</td>
<td>High</td>
<td>W3</td>
</tr>
<tr>
<td>12 or more</td>
<td>Very high</td>
<td>W4</td>
</tr>
</tbody>
</table>

**TABLE 5: COMPLEXITY AND WEIGHT FOR CR**

The set \( w_1, w_2, w_3 \) and \( w_4 \) may be different for program in different languages.

**3. FUZZIFICATION:**
The complexity attributes low, medium and high for the four metrics ALV, ALS, ACC, and CR are taken as triangular fuzzy number (TFN). membership functions are evaluated using complexity and coefficient matrices. Fuzzy pictorial representation of TFNs for ALV, ALS ACC and CR are shown in Figure:

![Fuzzy Pictorial Representation of ALV](image-url)
4. Defuzzification:
Defuzzification process is applied to evaluate crisp value of maintainability factors for ALV,ALS,ACC and CR. 

R and Rr are used to represent corresponding factors of maintainability for ALV,ALS,ACC and CR respectively. Defuzzification rules are defined in the following equations.

\[
R_{\text{ALV}} = \begin{cases} 
\mu * w_1 & 0 < ALV \leq 1 \\
\mu * w_1 + (1 - \mu) * w_2, & 1 < ALV \leq 2 \\
\mu * w_2 + (1 - \mu) * w_2, & 2 < ALV \leq 3.5 \\
\mu * w_2 + (1 - \mu) * w_3, & 3.5 < ALV \leq 5 \\
\mu * w_3 + (1 - \mu) * w_3, & 5 < ALV \leq 6.5 \\
\mu * w_3 + (1 - \mu) * w_4, & 6.5 < ALV \leq 8 \\
\frac{w_4}{ALV > 8} & 
\end{cases}
\]

\[
R_{\text{ACC}} = \begin{cases} 
\mu * w_1 & 0 < ACC \leq 2.5 \\
\mu * w_1 + (1 - \mu) * w_2, & 2.5 < ACC \leq 5 \\
\mu * w_2 + (1 - \mu) * w_2, & 5 < ACC \leq 9 \\
\mu * w_2 + (1 - \mu) * w_3, & 9 < ACC \leq 13 \\
\mu * w_3 + (1 - \mu) * w_3, & 13 < ACC \leq 15.5 \\
\mu * w_3 + (1 - \mu) * w_4, & 15.5 < ALV \leq 18 \\
\frac{w_4}{ACC > 18} & 
\end{cases}
\]

\[
R_{\text{CR}} = \begin{cases} 
\mu * w_1 & 0 < CR \leq 2 \\
\mu * w_1 + (1 - \mu) * w_2, & 2 < CR \leq 4 \\
\mu * w_2 + (1 - \mu) * w_2, & 4 < CR \leq 6 \\
\mu * w_2 + (1 - \mu) * w_3, & 6 < CR \leq 8 \\
\mu * w_3 + (1 - \mu) * w_3, & 8 < CR \leq 10 \\
\mu * w_3 + (1 - \mu) * w_4, & 10 < CR \leq 12 \\
\frac{w_4}{CR > 12} & 
\end{cases}
\]

Where \( \mu \) represents grade of membership function. Since maintainability of software is inversely proportional to ALV,ALS,ACC and CR respectively. The following equation used to evaluate maintainability

\[ \text{Maintainability} = R_{\text{ALV}} \times R_{\text{ALS}} \times R_{\text{ACC}} \times R_{\text{CR}} \]

5. Experimental Results:
In order to measure the performance of our system, we have used a published dataset of nine classroom projects[AGGA05]. Repair rate is considered for validation of results from present fuzzy logic based model. Repair rate is taken as inverse of average corrective maintainence time.

\[ \text{Repair rate} = \frac{1}{\text{Average CMT}} \]

Where Average CMT=average Corrective Maintenance Time. We considered the following values for weight factors for different levels of complexity.

\[ W_1=0.5, W_2=0.4, W_3=0.3 \text{ and } W_5=0.2 \]
Table 6: Validation Results for Maintainability Assessment Model.

<table>
<thead>
<tr>
<th>P.NO</th>
<th>ALV</th>
<th>ALS</th>
<th>ACC</th>
<th>CR</th>
<th>Maintainability</th>
<th>Repair rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.5</td>
<td>132</td>
<td>11.5</td>
<td>7.74</td>
<td>0.083</td>
<td>0.062</td>
</tr>
<tr>
<td>2</td>
<td>1.58</td>
<td>43.8</td>
<td>12.6</td>
<td>5.62</td>
<td>0.149</td>
<td>0.065</td>
</tr>
<tr>
<td>3</td>
<td>4.41</td>
<td>238</td>
<td>5.28</td>
<td>8.30</td>
<td>0.115</td>
<td>0.056</td>
</tr>
<tr>
<td>4</td>
<td>3.95</td>
<td>292</td>
<td>13.7</td>
<td>8.80</td>
<td>0.066</td>
<td>0.047</td>
</tr>
<tr>
<td>5</td>
<td>2.32</td>
<td>118</td>
<td>7.43</td>
<td>7.32</td>
<td>0.135</td>
<td>0.067</td>
</tr>
<tr>
<td>6</td>
<td>3.14</td>
<td>288</td>
<td>10.7</td>
<td>9.23</td>
<td>0.071</td>
<td>0.056</td>
</tr>
<tr>
<td>7</td>
<td>3.14</td>
<td>141</td>
<td>9.37</td>
<td>6.89</td>
<td>0.088</td>
<td>0.058</td>
</tr>
<tr>
<td>8</td>
<td>5.86</td>
<td>298</td>
<td>7.0</td>
<td>7.00</td>
<td>0.076</td>
<td>0.045</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>300</td>
<td>10.7</td>
<td>8.80</td>
<td>0.056</td>
<td>0.040</td>
</tr>
</tbody>
</table>

The graph of maintainability of nine projects under study from proposed model is compared which shows a strong correlation between integrated measure of maintainability with maintenance time.
6. Conclusion:

In this paper a method is proposed for predicting the maintainability of a software application based on the combined effect of four important factors: ALV, ALS, ACC, and CR. This integrated approach gives a true picture of the software maintainability. The model is validated on select software projects to check the usefulness of the approach. The maintainability result of the proposed model shows excellent correlation with repair rate, where as individual factors show a little correlation with the same. Lower value of maintainability indicates the need for improvement in software, so that the maintenance cost can be reduced. A crisp value for maintainability can help software managers to judge the maintenance efforts required for various maintenance activities. Further the measured value can be used to estimate changeability and reusability. The technique of fuzzy logic used in study is sufficiently general and can be easily applied to other areas of quantitative software engineering.

7. REFERENCES


