CHAPTER 8
MEASUREMENT OF SOFTWARE MAINTAINABILITY

8.1 INTRODUCTION

Every software needs to be modified to meet customer's requirement in its life cycle. Software maintenance is a task that every development group has to face when the software is delivered to the customers' site, installed and is operational. The time spent and effort required for keeping software operational consumes about 40 - 70% of cost of entire life cycle.

Software maintenance encompasses a broad range of activities including error corrections, enhancement of capabilities, deletion of obsolete capabilities and optimization. [103,104] The value of software can be enhanced by meeting additional requirements, making it easier to use, more efficient and employing newer technologies. [82] Maintenance may span for fifteen years whereas development may be 1-2 year [3,11]. Even though it is an important task, it is poorly managed. The fact that you cannot control, what you cannot measure, makes measurement of maintainability very important. Software maintenance is defined as process of modifying existing operational software while leaving its
primary functions intact [32] of maintenance [33] includes the following activities:

- Redesign and redevelopments of small portions of existing software
- Design and development of small interfacing software packages, which require some redesign of the existing software product.
- Modification of software product's code, documentation and data base structure.

However it excludes the following:

- Major redesign & redevelopment (more than 50%)
- Design and development of a sizeable (more than 20% of source instructions of existing product) interfacing software package.

Data processing system operations, data entry and modification of values in the database.

Software Maintenance can be classified as:

- Software update which results in changed final specification for software specification for software product
- Software repair where final specification is not changed.
8.2 EXISTING MODELS

In literature, some metrics have been proposed for measuring / predicting maintainability. In 1984 a tool was proposed [33] which operated at syntactic level. Then another model [228] was proposed which considered design attributes. Another model was proposed [225] which used quality metrics. Software Maturity index [SMI] [116] considered only modules being added or removed.

Another model [171] was proposed which considers only the design aspect. This model does not take into account the quality of documentation or understandability of source code. The authors have admitted that the model may not be suitable in all environments as maintainability depends on many more factors other than design characteristics. In [160], a set of metrics was identified forming a hierarchical tree structure of around fifty metrics. But no measurement technique for most of these was proposed. The formula proposed [160] used different weights for each of individual metrics but no weights were defined. These weights are subjective in nature and subjectivity can be easily handled with the help of fuzzy logic.
A fuzzy model [17] has been proposed where Maintainability is a measure of characteristics of software e.g. source code readability, documentation quality and cohesiveness among source code and documents.

It is also seen that maintainability very much depends on the average number of live variables in a program and average life span of variables. Presently there is no model that considers the effect of these two factors. Therefore, a model which integrates the four factors namely average number of Live Variables $LV$, average Life Span ($LS$) of variables, the average Cyclomatic Complexity ($ACC$)[27,83] and the Comments Ratio ($CR$) and provides a measure of maintainability, is proposed.

**8.3 FACTORS AFFECTING MAINTAINABILITY**

**8.3.1 AVERAGE NUMBER OF LIVE VARIABLES**

A live variable is live at a particular statement only if it is referenced a certain number of statements before or after that statement. The average number of live variables ($LV$) is the sum of the count of Live Variables divided by the count of executable statements ($n$) [8]

$$LV = \frac{LV}{n}$$

For a program having $m$ Modules $LV_{\text{program}} = \frac{\sum_{i=1}^{m} LV_{i}}{m}$ \hspace{1cm} \ldots ..(8.1)
The more, the average number of live variables, the more difficult it would be to develop and to maintain a software.

8.3.2 AVERAGE LIVE VARIABLE SPAN

The span is the number of statements between two successive references of the same variable [8, 71]. The average span size (LS) for a program of n spans could be computed using the equation

$$\bar{LS}_{\text{program}} = \frac{\sum_{i=1}^{n} LS_i}{n} \quad \ldots \ldots \text{(8.3)}$$

8.3.3 COMMENTS RATIO

Comment ratio [64] is defined as

$$CR = (s+c)/c \quad \ldots \ldots \text{(8.4)}$$

Where s denotes total lines of code and c represents total number of comment lines. The lower the ratio, the better is the readability, and the better the readability, the better is the maintainability. Comments provide better readability and therefore the Comments Ratio is an important factor that affects maintainability.
8.3.4 AVERAGE CYCLOMATIC COMPLEXITY

McCabe [157] has defined Cyclomatic Complexity as

\[ V = e - n + 2p. \] ...........(8.4)

Where \( e \) is the number of edges in a program flow graph, \( n \) the number of nodes and \( p \) the number of connected components. If \( p = 1 \), then \( v = \Pi + 1 \) where \( \Pi \) is the number of predicates in the program. The Average Cyclomatic Complexity (ACC) is defined as average of cyclomatic complexities of all modules.

8.4 PROPOSED FUZZY MODEL

There are four inputs to the fuzzy model, namely Average Live Variables, Average Life Span of variables, Average Cyclomatic Complexity and Comments Ratio. Figure 8.1 shows the Fuzzy Model.
This model considers all four inputs and provides a crisp value of maintainability using the Rule Base. All inputs can be classified into fuzzy sets viz. Low, Medium and High. The output maintainability is classified as Very Good, Good Average, Poor and Very Poor.

In order to fuzzify the inputs, the following membership functions are chosen namely Low, Medium and High. They are shown in Figures 8.2 to 8.5 as shown below
Figure 8.2: Fuzzification of Average live variable.

Figure 8.3: Fuzzification of Average Life Span

Figure 8.4: Fuzzification of Average Cyclomatic Complexity
Figure 8.5: Fuzzification of Comment Ratio

Similarly the output variable i.e. maintainability has five membership functions as shown in Figure 8.6.

Figure 8.6: Fuzzification of Output Variable – maintainability

8.5 EXPERIMENTAL METHODOLOGY

(i) All the inputs and outputs were fuzzified as shown in figures 8.2 to 8.6.

(ii) All possible combination of inputs were considered which leads to $3^4$ i.e. 81 sets.
The maintainability in case of all eighty-one combinations is classified as Very Good, Good, Average, Poor or Very Poor by expert opinion. These lead to formation of 81 rules for the fuzzy model and some of them are shown below:

1. If (\textit{CR} is low) and (\textit{ACC} is low) and (\textit{LV} is low) and (\textit{LS} is low) then maintainability is very good.

2. If (\textit{CR} is low) and (\textit{ACC} is low) and (\textit{LV} is low) and (\textit{LS} is med) then maintainability is very good.

81. If (\textit{CR} is high) and (\textit{ACC} is high) and (\textit{LV} is high) and (\textit{LS} is high) then maintainability is very poor.

(iii) All eighty-one rules are inserted and a rule base is created. Depending on a particular set of inputs, a rule will be fired.

(iv) Mamdani style of inference is used.
(v) Using the rule viewer, output i.e. maintainability is observed for a particular set of inputs using the MATLAB Fuzzy toolbox.

(vi) The output is also calculated theoretically using the Center of gravity.

8.6 RESULTS

Output computation for the model:

Let us say we have the following inputs to the model.

\[ ACC = 2, CR = 12, LV = 1, LS = 130 \]

When those inputs are fuzzified we find that \( ACC = 2 \) belongs to fuzzy set low with membership grade 1, \( CR = 12 \) belongs to fuzzy set low with membership grade 1, \( LV = 1 \) belongs to fuzzy set low with membership grade 1 and \( LS = 130 \) belongs to fuzzy set low with membership grade 0.25 and medium with membership grade 0.5. With these inputs, rule number 1 and 2 fire. During composition of these rules we get the following

\[ \text{Min}(1, 1, 1, 0.25) = 0.25 \]

\[ \text{Min}(1, 1, 1, 0.5) = 0.5. \]

When these two rules are implicated, we find that the first rule gives maintainability very good to an extent of 0.25 and second rule gives the maintainability value good to the extent of 0.5. This is shown in Figure 8.7.
Figure 8.7: Output Computation of Maintainability

\[
\text{Maintainability} = \frac{\int_{0}^{5} \frac{1}{2.5} x \, dx + \int_{3}^{5} (mx+c)x \, dx + \int_{0}^{0.5} 0.5x \, dx + \int_{0}^{5} (mx+c)x \, dx}{\int_{0}^{5} \frac{1}{2.5} x \, dx + \int_{3}^{5} (mx+c)x \, dx + \int_{0}^{0.5} 0.5x \, dx + \int_{0}^{5} (mx+c)x \, dx}
\]

\[
= \frac{\int_{0}^{5} \left( \frac{0.25x^2}{2} \right) \, dx + \left[ \frac{0.5x^3}{3} - \frac{x^2}{2} \right]_{3}^{5} + \left[ \frac{0.5x^2}{2} \right]_{0}^{0.5} + \left[ \frac{-0.5x^3}{3} + \frac{3x^2}{2} \right]_{0}^{5}}{\int_{0}^{5} \left( \frac{0.25x^2}{2} \right) \, dx + \left[ \frac{0.5x^3}{3} - \frac{x^2}{2} \right]_{3}^{5} + \left[ 0.5x \right]_{0}^{0.5} + \left[ \frac{-0.5x^2}{2} + 3x \right]_{0}^{5}}
\]

= 3.2

Defuzzification:

Defuzzification of the above output can be obtained by finding the Centre of Gravity [121] of the above fuzzy output.
The effect of these rules was observed also by simulating the model in MATLAB Fuzzy Tool Box. The maintainability for the above-mentioned inputs comes out to be 3.2, which is the same as calculated above. The various surface views of the simulated model are shown in Figure 8.8 & Figure 8.9.

Figure 8.8: Surface view with $\overline{LS}$ input as x-axis and $ACC$ taken on y-axis and maintainability on z-axis,
Figure 8.9: Surface view with $LV$ input as x-axis and ACC taken on y-axis and maintainability on z-axis

8.7 EMPIRICAL VALIDATION

In order to validate the model, ten procedure oriented software projects of undergraduate engineering students were considered. They were chosen only when proper set of input variables was available. Some logical errors were introduced in these projects and time take for corrective maintenance action was measured (Average. CMT). The maintainability was also calculated using the proposed fuzzy model. The results are shown in Table 8.5
<table>
<thead>
<tr>
<th>Project No.</th>
<th>ACC</th>
<th>CR</th>
<th>$LV$</th>
<th>$LS$</th>
<th>Maint</th>
<th>Avg. CMT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>8.51</td>
<td>3.92</td>
<td>0.5</td>
<td>25.4</td>
<td>6</td>
<td>17.0</td>
</tr>
<tr>
<td>2.</td>
<td>11.5</td>
<td>7.74</td>
<td>2.5</td>
<td>132</td>
<td>5.39</td>
<td>16.10</td>
</tr>
<tr>
<td>3.</td>
<td>12.6</td>
<td>5.62</td>
<td>1.59</td>
<td>43.8</td>
<td>4.8</td>
<td>15.4</td>
</tr>
<tr>
<td>4.</td>
<td>5.28</td>
<td>8.30</td>
<td>4.41</td>
<td>238</td>
<td>6.87</td>
<td>18.0</td>
</tr>
<tr>
<td>5.</td>
<td>13.7</td>
<td>8.8</td>
<td>3.95</td>
<td>292</td>
<td>7.93</td>
<td>21.10</td>
</tr>
<tr>
<td>6.</td>
<td>7.43</td>
<td>7.32</td>
<td>2.32</td>
<td>118</td>
<td>4.49</td>
<td>15.0</td>
</tr>
<tr>
<td>7.</td>
<td>10.7</td>
<td>9.23</td>
<td>3.14</td>
<td>288</td>
<td>6.49</td>
<td>17.90</td>
</tr>
<tr>
<td>8.</td>
<td>9.37</td>
<td>6.89</td>
<td>3.14</td>
<td>141</td>
<td>6.49</td>
<td>17.20</td>
</tr>
<tr>
<td>9.</td>
<td>7.0</td>
<td>7.0</td>
<td>5.86</td>
<td>298</td>
<td>8</td>
<td>22.0</td>
</tr>
<tr>
<td>10.</td>
<td>10.7</td>
<td>8.8</td>
<td>6</td>
<td>300</td>
<td>10.5</td>
<td>25.2</td>
</tr>
</tbody>
</table>

Table 8.5  Values of maintenance time and maintainability

The values of average corrective maintenance time of these projects have been plotted against each of the four input metrics namely $LV$, $LS$, $ACC$ & $CR$ in Figures 8.10, 8.11, 8.12 & 8.13 respectively.
Figure 8.10: Average Live Variable vs. Average Corrective Maintenance Time.

Figure 8.11: Average Life Span vs. Average Corrective Maintenance Time
Figure 8.12: Average Cyclomatic Complexity vs Average Corrective Maintenance Time

Figure 8.13: Comment Ratio vs Maintenance Time
It can be seen there is hardly any co-relation between average maintenance time and the four inputs. These four metrics cannot individually predict the maintenance time. On the other hand a plot of maintainability versus maintenance time is shown in Figure 8.14. This shows that integrated measure of maintainability is strongly co-related with maintenance time.

Thus the fuzzy model is validated and that the integrated value of maintenance gives better results than any individual input metric is also verified with the help of empirical results.
8.8 CONCLUSION

Maintainability can be estimated with the help of fuzzy model and the empirical results prove that the integrated measure of maintenance obtained from this model shows a strong co-relation to the maintenance time. The four input metrics individually cannot individually predict the maintenance time.