CHAPTER 9

MEASUREMENT OF SOFTWARE UNDERSTANDABILITY

9.1 INTRODUCTION

Change is evident in every walk of life and softwares are no exception. During its life cycle, changes are made to it to meet user requirements. These changes in the software after delivery are known as software maintenance [207,211]. This scheme requires a deep understanding of the working and functionality of the software.

Understandability is the measure used for measuring the programmers' competence in understanding the software code and its associated documents.[210] Understandability is an important aspect, but is poorly managed and it is so because of the lack of a good measure of software understandability. The concept of life that "you can't control what you can't measure", holds true for good measurement of software understandability also [70].

Software maintenance is one of the most important aspects of software life cycle. The ever changing needs of the user, introduction of new hardware etc. makes maintenance all the more necessary. Software maintainability introduces concepts like good readability and understandability of source code as well as its
associated documents. This chapter introduces a neural network based model for measuring software understandability dependent on three parameters namely Total Spatial Complexity, Documentation Quality and Cohesiveness between source and the associated documentation.

9.2 EXISTING MEASURES OF SOFTWARE UNDERSTANDABILITY

Till date very few measures of understandability [64] have been proposed in literature or research papers. It has been proved that, comments improve the readability. Measure of understandability of source code was proposed in the form of spatial complexity [16, 70], which measures the spatial abilities needed to understand the source code.

Similarly the readability of the documentation was measured with the help of Fogs Gunning Index. Laitnen suggested a tool to measure the understandability of software documents [146]. It is based on the use of symbols in source code and other documents.

Understandability of software should be an integrated measure of software as a whole i.e. it should be a measure of software code, documentation quality and the cohesiveness among them.
associated documents. This chapter introduces a neural network based model for measuring software understandability dependent on three parameters namely Total Spatial Complexity, Documentation Quality and Cohesiveness between source and the associated documentation.

9.2 EXISTING MEASURES OF SOFTWARE UNDERSTANDABILITY

Till date very few measures of understandability [64] have been proposed in literature or research papers. It has been proved that, comments improve the readability. Measure of understandability of source code was proposed in the form of spatial complexity [16, 70], which measures the spatial abilities needed to understand the source code.

Similarly the readability of the documentation was measured with the help of Fogs Gunning Index. Laitnen suggested a tool to measure the understandability of software documents [146]. It is based on the use of symbols in source code and other documents.

Understandability of software should be an integrated measure of software as a whole i.e. it should be a measure of software code, documentation quality and the cohesiveness among them.
9.3 FACTORS AFFECTING UNDERSTANDABILITY

Understandability is dependent on the source code, its documentation and its interrelationship. This chapter suggests a neural network model, which considers the above inputs namely Total Spatial Complexity (TSC), the Documentation Quality (DOQ) and the Cohesiveness of Source Code and Documents (CSD). A neural network model for the above has been proposed [6].

9.3.1 TOTAL SPATIAL COMPLEXITY

The summation of CSC and DSC gives us the TSC.

\[ \text{i.e. } TSC = \text{CSC} + \text{DSC} \]  \hspace{1cm} (9.1)

Code Spatial Complexity (CSC) is a measure for understanding of the source code. It also depends on the psychological complexity of the source code.

Data Spatial Complexity (DSC) is a measure of the cognitive abilities to comprehend in the various inputs, output and other intermediate results. TSC is an important factor contributing to understandability because of these two factors namely CSC and DSC.
9.3.2 DOCUMENTATION QUALITY

The document quality associated with the code is of the same importance as the code itself. One of the well-known metrics for measuring documentation quality is Gunning's Fog Index, which is the measure of the readability of a passage of text [178]. The lower values ascertain the quality whereas higher values denote poor documentation quality.

9.3.3 COHESIVENESS OF SOURCE CODE AND DOCUMENTS (CSD)

Software includes both the source code and associated documents; therefore an important factor contributing to the understand ability of software is cohesiveness of source code and documents. A good co-relation between the code and its related document shows better understand ability; as the software developer can co-relate between the two.

As proposed by Laatmen [146]. If the language of the code and the documents is closely related, then such practice leads to better understand ability and lower effort for maintenance.
9.4 PROPOSED MODEL

The software understandability is dependent on three important parameters namely Total Spatial Complexity, the Documentation Quality which is measured by Fog’s Index and Cohesiveness of Source Code and Documentation. Earlier a Fuzzy model has been proposed for measuring the software understandability [16]. In the current work, we have tried to employ an artificial neural network based model for software understandability.

Neural networks have a capability to complex non-linear relationships and approximating any measurable function. They provide attributes, which make it a productive mechanism tool for pattern classification and clustering. Figure 9.1 shows a neural network model.

![Neural network model](image)

**Figure 9.1:** A neural network model to measure understandability
9.5 EXPERIMENTAL METHODOLOGY

The three principal factors affecting the software understandability are:

1. Total Spatial Complexity (TSC)
2. Documentation Quality (DQ)
3. Cohesiveness of Sources Code and Document (CSD)

These factors are considered as input to neural network and the target output that is generated is understandability. These three inputs can be low, medium or high. The ranges as classified by expert opinion are as shown in table 9.1. Then the inputs are normalized to have values between 0 and 1. All possible combinations of the three inputs are then considered. These combinations resulted in twenty-seven distinct cases.

Using MATLAB neural network tool, the network is trained for 65 different combinations resulting in diverse understandability values. The training inputs are then applied in numeric format. For example when TSC is high, it can be given any value between 0 to 0.33. Thus assigning numeric values to the twenty-seven distinct cases could lead to a number of training examples. A set of 65 exemplars was chosen for training, making certain that border values are included. In this model, the first step is to feed the system with a set of data (input-output information) and in the process train the network, second step constitutes
of adjustments to make outputs closer to what it should be. The adjustment is
called back propagation. The inputs as classified by expert opinion are as shown
in Table 9.1

<table>
<thead>
<tr>
<th>Factor</th>
<th>High</th>
<th>Medium</th>
<th>Low</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Normalized Values)</td>
<td>0-0.33</td>
<td>0.34-0.67</td>
<td>0.68 - 1</td>
</tr>
<tr>
<td>DOQ</td>
<td>&lt;12</td>
<td>12-18</td>
<td>18-25</td>
</tr>
<tr>
<td>(Normalized Values)</td>
<td>0 - 0.48</td>
<td>0.49-0.72</td>
<td>0.73-1</td>
</tr>
<tr>
<td>CSD</td>
<td>&lt;500</td>
<td>500-900</td>
<td>900-1500</td>
</tr>
<tr>
<td>(Normalized Values)</td>
<td>0 - 0.33</td>
<td>0.34 - 0.66</td>
<td>0.6 - 1.1</td>
</tr>
</tbody>
</table>

Table 9.1: Inputs as classified by expert opinion

The understandability i.e. the target output of the neural network was categorized
into four types as shown in Table 9.2

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Low</td>
<td>U1</td>
</tr>
<tr>
<td>01</td>
<td>Medium</td>
<td>U2</td>
</tr>
<tr>
<td>10</td>
<td>High</td>
<td>U3</td>
</tr>
<tr>
<td>11</td>
<td>Very High</td>
<td>U4</td>
</tr>
</tbody>
</table>

Table 9.2: Classification of understandability

The error back propagation is used to perform the experiment. A sigmoidal feed
forward network with a single hidden layer is used. There are three nodes in the
input layer. In the first (hidden) layer neurons are varied from 2 to 23 and in the
second (output) layer 2 neurons are kept to represent the four target values as
shown in Table 9.2
Trainlm function of MATLAB is used for training and adaptive learning function selected is 'Learnngdm' Performance function was measured by mean square error 'MSE'. Transfer function is formulated by 'log sigmoid' in both the layers.

The network with 22 neurons in the hidden layer is found to be most appropriate for further study, as it gave the best results. Taking the network with 22 hidden neurons, experimentation is done with the same parameter values. The epochs considered are 1000 and the goal is kept as 0.05.

The training set of 65 exemplars consists of 20 each from U1, U2 and U3 categories and 5 from U4 category. These were chosen randomly. After the training, results are found to be satisfactory, and then simulation is done on the network. Misclassification is a situation when the measured values differ i.e. if the measured value is set to (0, 0) then any other value except (0, 0) is a misclassification. The values of inputs are rounded off for example values from 0-0.35 are treated a zero, similarly values from 0.65-1 are treated as 1. Test data for 5 sets of 57 combinations each is provided as inputs to the neural network, which is already training with a goal of 0 - 0.05.

9.6 RESULTS & DISCUSSION

The results are as shown below for the experiment
<table>
<thead>
<tr>
<th>No of training cases</th>
<th>Misclassification</th>
<th>Misclassification</th>
<th>% Misclassification</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Percentage</td>
<td>combined</td>
</tr>
<tr>
<td></td>
<td>U1    U2 U3 U4</td>
<td>U1    U2 U3 U4</td>
<td>(U1, U2, U3, U4)</td>
</tr>
<tr>
<td>20</td>
<td>0     3  2 0</td>
<td>0     15 10 0</td>
<td>6.02%</td>
</tr>
</tbody>
</table>

Table 9.3: Training results (65 cases)

<table>
<thead>
<tr>
<th>Misclassification (Mean)</th>
<th>Misclassification (Std. Deviation.)</th>
<th>Percentage of Misclassification</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1       U2 U3 U4</td>
<td>U1       U2 U3 U4</td>
<td>U1     U2 U3 U4</td>
</tr>
<tr>
<td>2.2       6.4 2.6 0.2</td>
<td>1.1      3.2 2.4 0.8</td>
<td>13.75 40 16.2 1.25</td>
</tr>
</tbody>
</table>

Table 9.4: Simulation Results (5 sets of test data of 57 cases each)

<table>
<thead>
<tr>
<th>Misclassification (Mean)</th>
<th>Misclassification (Std. Deviation.)</th>
<th>Percentage of Misclassification</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1       U2 U3 U4</td>
<td>U1       U2 U3 U4</td>
<td>U1     U2 U3 U4</td>
</tr>
<tr>
<td>4         5  3 0</td>
<td>1.2      2.2 1.3 0</td>
<td>8      10.4 6.25 0</td>
</tr>
</tbody>
</table>

Table 9.5: Simulation Results (5 sets of test data of 171 cases each)

Simulation is done again with 171 test cases to use the difference in classification (if any), if it is found that the difference is significant. In case of simulation done with 57 exemplars the misclassification is found to be 17.4% where as in case of 171 cases misclassification is reduced to 6.16%.
The network was trained well and the percentage of misclassification cases observed was 6.02, that too mainly in the U2 category.

After the network is simulated with sets of 57 exemplars each, percentage of maximum misclassification is observed in U2 category with standard deviation of 3.2. Average percentage of misclassification is 17.4 in all four categories. Again maximum misclassification was observed in the U2 category (10.5%) with standard deviation as 3.2 and the percentage of misclassification is 6.16 in all four categories. With 171 exemplars the misclassification improved considerably.

Thus with more number of exemplars used in simulation the result show improvement and the misclassification cases reduce to a considerable extent.

9.7 CONCLUSION

In this Chapter, software understandability is proposed as an integrated measure of mainly three characteristics of the software i.e. Total Spatial Complexity, Documentation Quality and Cohesiveness of Source Code and Data Documents. This integration is done with the help of a neural network, which can handle the subjectivity of their individual contribution toward understandability. Thus it is observed from the experimental work, that neural network can be very well used for this purpose.