# CHAPTER – 5

## VALIDATION OF PROPOSED METRICS

<table>
<thead>
<tr>
<th>S. No.</th>
<th>Contents</th>
<th>Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Metrics and Their Validation</td>
<td>85</td>
</tr>
<tr>
<td>5.2</td>
<td>Evaluation of Software Metrics – Various Criteria</td>
<td>86</td>
</tr>
<tr>
<td>5.3</td>
<td>Discussion</td>
<td>87</td>
</tr>
<tr>
<td>5.4</td>
<td>Metrics Validation</td>
<td>88</td>
</tr>
<tr>
<td>5.5</td>
<td>Empirical Validation</td>
<td>92</td>
</tr>
<tr>
<td></td>
<td>5.5.1 RFV case study</td>
<td>92</td>
</tr>
<tr>
<td></td>
<td>5.5.2 JLex case study</td>
<td>98</td>
</tr>
<tr>
<td></td>
<td>Summary</td>
<td>99</td>
</tr>
</tbody>
</table>
CHAPTER – 5
VALIDATION OF PROPOSED METRICS

EveryMetric has to be evaluated to prove that it is applicable for its intended purpose. With this intention validating metrics that are proposed in the previous chapter is considered as the next task of the methodology. So, this chapter focuses on presenting validation, importance of validation, types of validation, various methods that are available for validation. A comparative study has been provided in literature survey to find out the best possible way for validating metrics. With enough study with this perspective it has been concluded that Weyuker’s principles are the best way to validate the metrics. Hence, proposed metrics are validated using Weyuker properties as a part of analytical validation. To validate the proposed metrics suit empirically two open source softwaresystems, RFV and JLEX have been considered.

5.1 METRICS AND THEIR VALIDATION

Software engineering is a pragmatic branch; here empirical terms such as maintainability and understandability are experienced by software engineers [118]. Here these terms have to be coined in to form of generally acceptable attributes which should be confined by software metrics.

Measures and way of measuring in software engineering is different from that of other disciplines of engineering. In other branches of engineering measurements are based upon some basic principles like Ohm’s law, Newton’s law etc. For instance, when an engineering product is considered, using some formula product’s size can be easily measured but obviously the size that has been measured cannot be altered. On the other hand, to measure size of the software such as program length no standards are available. It is not clear, whether blank lines, irrelevant lines, comment lines etc., should be considered or not. Representation is another problem, it is related to proper representation
of empirical statements by numbers. In software measurement area another problem is uniqueness. It deals to check whether the transformation of the number is without any change in empirical denotation [118]. For instance, there are two diverse formulas for same empirical denotation in McCabe’s metrics and hence, their numerical way of representation is not the same. Another difficulty of this area is meaningfulness, it states meaningfulness of statements regarding various kinds of transformation. Meaningfulness has been identified as very important in three regions of software measurement, they are statistical operations, combinational rules and dimensional analysis, according to Zuse [118]. Along with these difficulties there are many other troubles that can be observed in measuring software, for instance, when a complexity measure is considered proposing pragmatic relation system for it is not simple i.e., acquiring empirical relations between entities. In the life of human beings, establishing relationships between entities is easy when compared to establishing such types of relations between software [223] for instance “A” is taller to “B”, which is not possible between software. Many researchers are contributing to lay a foundation for measuring software, examples can be found in [224]. All these contributions can be categorized into two groups, one is to regulate the fundamental rules for measuring software and other group is developing various kinds of techniques for measuring software. Weyuker’s [225], Zuse’s [226], Kitchenhan’s [227], Braind’s [126,228], Wangs’s [224] and Kaner’s [229] works do come under first category.

5.2 EVALUATION OF SOFTWARE METRICS – VARIOUS CRITERIA

Numerous formal frameworks have been produced by many scientists for the purpose of validating the software metrics. There are basically two categories of techniques for validation. The empirical validation of the proposed metric which confirms its actual applicability is the most dominant of these two techniques. However, prior to the empirical validation i.e., applying to industry, every metric has to be analytically evaluated to confirm that it is having scientific pedestal, and it has been developed based
on definite measurement theories. Many authors formulated properties with respect to software metrics. Some of them are Bache [118], Ejiogu [127], Conte et al. [124], Ions [123], Kearney [125], Sheppard and Ince [32], Watts [122], Zuse [226] and Weyuker [225].

All the above stated works have been developed in order to cover all the features of metrics from various perspectives. No uniformity is observed among all these metrics, some of these metrics are either theoretical based properties that are subjective such as properties that are proposed by Conte et al. [124] or mathematical properties such as the properties that are proposed by Bache. Nine properties proposed by Bache are associated with programs flow graph depiction, later flow graph has been described in a special manner. It has been stated by Conte et al. [124] that algorithms complexity should not differ with place, time or observer. Moreover, some theoretical properties of the measure such as simplicity, also have been proposed by them. Compared with all these properties the desiderated proposed by Weyuker have attained more popularity. These are most extensively used for analytical evaluation of metrics and as a proof for their usage research papers in hundreds are available.

5.3 DISCUSSION

If an attribute is able to differentiate one entity from another then it is a measurable attribute. In Weyuker’s list of properties, the first property affirms that when a metric is applied over two entities it should result in two different values. Ninth property of Weyuker is related with representation condition, which must be obeyed by a valid metric that is it must conserve the instinctive conceptions regarding the attribute along with the manner in which it discriminates different entities. Representation condition implies an assertion stating that complexity is related to the structural complexity but not to the psychological complexity. Each and every element of the contributing attribute to the valid metric must be equivalent. Weyuker’s seventh property is related with this issue.

A metric is said to be valid when it produces same value over different artifacts. This property is logically required. In further domains also this property is reliable with value of other aspects. Hence, this property is considered as a common property of a metric.
According to Weyuker’s fourth property, artifacts with same functionality may be having different values for the metric. This property affirms that the function will not stipulate form. This property is also be considered as a common property. On similar lines Weyuker’s second and third properties are also be considered as common properties of a metric. Monotonicity is the fifth one among Weyuker’s properties, this property is related to composition this is associated with one of the extensive structure axioms. The perspective that is reflected by this property is, the important factor in accessing program’s complexity is numeral of structural components present in that program. According to sixth property of Weyuker comprehensibility is the major factor that influences the program’s or artifact’s complexity and only a little amount of prominence is given to structural components of the program. Customized ninth property of Weyuker is the salient property. Along with confirming complexity’s additive nature it also confirms that the metric should be on ratio scale. This property is also associated with the wholeness which have to be significant with the ratio scale. Furthermore, all the Weyuker’s properties are the outcomes of bonafide experiments over software metrics; hence, the properties of metrics have been characterized in a realistic manner.

5.4 METRICS VALIDATION

Some important properties and validation criteria are available in the literature that is extensively used by various researchers that any of the proposed metrics must satisfy. Among numerous evaluation criteria that exist in the literature, Weyuker’s properties are most extensively used for evaluating software metrics. There are many unique features of Weyuker’s properties such as they have scientific base in their development, they are represented in terms of mathematical expressions, these properties are very clear and hence easy to understand. Weyuker’s properties are guiding tools for identifying good and complete metrics.

E.J.Weyuker has developed a formal set of desiderata for software metrics [230]. These properties are used to evaluate numerous existing software metrics. The notion of monotonicity, interaction, non-coarseness, non-uniqueness and permutation can be found in these desiderata. Generally, analytical evaluation is done by evaluating metrics against
the Weyuker’s principles. All the proposed metrics are validated as shown below and tabulated in table 5.1 that shows the metrics that satisfy the corresponding property.

**Property 1: Non-Coarseness**

A metric \( \mu \) when applied on artifacts \( A \) and \( B \) it states that \( \mu(A) \neq \mu(B) \). That is a metric should not produce the same value for all artifacts. If it does so it loses its value as a measurement.

Consider \( MVT \) metric. This metric satisfies the above property because it considers sizes of various types of variables. When two variables, a file \( v_1 \) and integer \( v_2 \) are considered then \( \mu(v_1) = 10/10 = 1 \) and \( \mu(v_2) = 1/10 = 0.1 \). Hence \( \mu(v_1) \neq \mu(v_2) \). Here, \( \mu \) refers to the corresponding metric in discussion.

Consider \( IM_i \) metric. This metric satisfies the above property as it measures the number of method’s instance accessing the variable within the module. When two variables \( v_1, v_2 \) are considered and \( v_1 \) is being accessed by 2 methods out of 5 methods, and \( v_2 \) is being accessed by 3 methods out of 5 methods. Then \( \mu(v_1) = 2/5 = 0.4 \) and \( \mu(v_2) = 3/5 = 0.6 \). Hence \( \mu(v_1) \neq \mu(v_2) \).

**Property 2: Granularity**

It states that there will be a finite number of cases for which the metric value will be same. As the universe deals with finite set of applications, a finite set of cases with the same metric value will be found. This property will be satisfied by any metric which is measured at variable level. In this thesis all the metrics proposed are measured at variable level and hence all of them satisfy this property.

**Property 3: Non-Uniqueness (Notion of equivalence)**

If \( A \) and \( B \) are distinct artifacts then the property states that \( \mu(A) = \mu(B) \). That is two artifacts may have same value for the metric which implies that both artifacts are equally complex.

\( MVT \) metric satisfies this property. When two variables float \( v_1 \), float \( v_2 \) are considered then \( \mu(v_1) = 2/10 = 0.2 \) and \( \mu(v_2) = 2/10 = 0.2 \). Hence \( \mu(v_1) = \mu(v_2) \).
Consider $IM_i$ metric. This metric satisfies the above property. When two variables $v_1, v_2$ are considered and both $v_1, v_2$ are being accessed by 2 methods out of 5 methods then $\mu(v_1)=2/5=0.4$ and $\mu(v_2)=2/5=0.4$. Hence $\mu(v_1) = \mu(v_2)$.

**Property 4: Design Details are Important**

This property states that, in determining the metric for an artifact its design details also matters. When designs of two artifacts A and B are considered which are same in functionality i.e., A=B does not imply that $\mu(A)=\mu(B)$. That means specifics of the artifact design must influence the metric value.

For instance $MVT$ satisfies this property. Consider designs of two methods A and B having same functionality (A=B). If $v_1$ is an integer variable in the design of module A and $v_2$ is a float variable in the design of module B then $\mu(v_1)=1/10=0.1$ and $\mu(v_2)=2/10=0.2$. Hence $\mu(v_1) \neq \mu(v_2)$

Consider $IM_i$ metric. This metric satisfies the above property. Let in design of module A variable $v_1$, is accessed by 3 methods and in design of module B by 2 methods out of 5 methods, then $\mu(v_1\text{ in } A)=3/5=0.6$ and $\mu(v_1 \text{ in } B)=2/5=0.4$. Hence $\mu(v_1 \text{ in } A) \neq \mu(v_2 \text{ in } B)$.

**Property 5: Monotonicity**

It states that a component of a program is always simpler than the whole program. For all A and B, $\mu(A)\leq\mu(A+B)$ and $\mu(B)\leq\mu(A+B)$ should hold. This implies that the metric for the combination of two artifacts can never be less than the metric for either of the artifacts.

As the type of a variable does not have any component, this property is not applicable for the metrics $MVT$ and $IM_i$.

**Property 6: Non-Equivalence of Interaction**

It states that two artifacts with the same complexity need not have the same complexity after being concatenated with a third artifact. When A, B and C are three artifacts and $\mu(A)=\mu(B)$ does not imply that $\mu(A+C)=\mu(B+C)$. This suggests that interaction between A and C may differ from that of B and C.
This property is not applicable for $MVT$. The metric depends only on the type of the variables and concatenation has no effect on the metric value. $IM_i$ satisfies this property. Because, methods can be concatenated and complexity of the concatenated methods will differ with that of the earlier one.

**Property 7: Permutation**

It states that permutation of elements within the artifact being measured may change the metric value. When there is an artifact $A$ and by permuting the order of the statements of $A$ artifact $B$ is formed then this property requires that $\mu(A) \neq \mu(B)$.

This property is not applicable for $MVT$. As the metrics are developed at variable level for $IM_i$ this property is not satisfied.

**Property 8: Renaming property**

When name of the measured artifact changes, the metric should not change. That is, if artifact $B$ is obtained by renaming artifact $A$ then $\mu(A) = \mu(B)$.

$MVT$ and $IM_i$ metrics satisfy this property as size of the variable and method interactions will not be changed when it is renamed.

**Property 9: Interaction Increases Complexity**

It states that when two artifacts are combined, interaction between them can increase the metric value. When two artifacts $A$ and $B$ are considered $\mu(A)+\mu(B) < \mu(A+B)$.

This property is not applicable for $MVT$. The metric depends only on the type of the variables so there is no interaction hence, no effect on the metric value. $IM_i$ metric will not satisfy this property because though complexity increases by combining the methods, the metric is defined at variable level and its complexity will not be affected.

Similarly, other metrics are validated and consolidated evaluation of all proposed metrics is tabulated in table 5.1. In the table, Y represents satisfied, N represents not satisfied and NA represents not applicable. It is clear from the table that the proposed metrics satisfy all the applicable Weyuker properties. Property 9 is not applicable or not satisfied by any of the proposed metrics.
Table 5.1: Metrics Evaluation Table for Variables

<table>
<thead>
<tr>
<th>Metric</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>P7</th>
<th>P8</th>
<th>P9</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVT</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>Y</td>
<td>NA</td>
</tr>
<tr>
<td>IM&lt;sub&gt;i&lt;/sub&gt;</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>NA</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>IM&lt;sub:o&lt;/sub&gt;</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>NA</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>IV&lt;sub&gt;i&lt;/sub&gt;</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>NA</td>
<td>NA</td>
<td>N</td>
<td>Y</td>
<td>NA</td>
</tr>
<tr>
<td>IV&lt;sub:o&lt;/sub&gt;</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>NA</td>
<td>NA</td>
<td>N</td>
<td>Y</td>
<td>NA</td>
</tr>
</tbody>
</table>

The proposed metrics are validated analytically against the Weyuker’s principles [54]. All applicable principles are satisfied by the proposed metrics. Hence, proposed metrics are effective and can be utilized for the purpose of categorization.

5.5 EMPIRICAL VALIDATION

Empirical validation is extremely significant for the accomplishment of any software measurement project (Basili et.al.1999; Feuton and Pfleeger, 1997; Kitcheuham et.al.,1995; Schneidewind, 1992). Metrics proposal do not have any value without demonstrating its practical utilization. Hence, the prime intention is to examine the applicability and usefulness of proposed metrics suite which has been already validated analytically for the purpose of variable categorization.

After completing analytical validation, to perform the empirical validation of the proposed work different software systems Restricted Focus Viewer (RFV) and JLex[55,56] are considered as casestudies. RFV is considered initially for the purpose of empirical validation then it is followed by the discussion regarding the software JLex.

5.5.1 RFV Case Study

Restricted Focus Viewer is a software system and it is now available in its latest version 2.1 with several new features, it provides more flexibility for the purpose of experimentation. This version replaces its older version i.e., version 1.1 which has only
13 classes, these were added by 22 new classes giving raise to new version 2.1 with altogether 35 classes in it. Huge change can be observed in version 2.1 when compared with version 1.1. Following are the steps that explain the procedure of empirical validation of RFV.

RFV validation is as follows.

- Source code of this software is given as input to the front end of Daikon tool and trace file is generated.
- Using Rational Rose software, source code is reverse engineered to obtain class diagrams.
- From these trace file and class diagrams, information regarding various interactions among the classes, methods, variables and type of variables are obtained.
- This information is used to compute the values for proposed metrics.
- Based on these values variables are categorized as design and non-design.

For RFV, using above procedure and values obtained for various metrics, all the variables of RFV 2.1 are categorized into design and non-design. Initially, RFV_Text_Line is considered, metrics are calculated and values are tabulated in table 5.2.

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Label</th>
<th>Font</th>
<th>Font_metrics</th>
<th>Color</th>
<th>Height_offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>$IM_i$</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>$IM_o$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$IV_i$</td>
<td>0.2</td>
<td>0.2</td>
<td>0.6</td>
<td>0.2</td>
<td>0.4</td>
</tr>
<tr>
<td>$IV_o$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>$MVT$</td>
<td>0.3</td>
<td>0.6</td>
<td>0.6</td>
<td>0.6</td>
<td>0.1</td>
</tr>
</tbody>
</table>

RFV_Text_Line has five variables namely Label, Font, Font_metrics, Color, Height_offset. Values of all the five proposed metrics are calculated for every variable and they are recorded in table 5.2. Variables are categorized as design and non-design.
variables. If the criterion is such that fifty percent of the metrics have a value that is equal to or greater than the threshold value (0.5), then they are categorized as design variables. Otherwise they are categorized as non-design variables.

For the variable Label, interaction metric of variable with methods inside the module, $IM_i$ value is 0.5, interaction metric of variable with methods outside the module, $IM_o$ value is 0, interaction metric of variable with other variables inside the module, $IV_i$ value is 0.2, interaction metric of variable with other variables outside the module, $IV_o$ value is 0, metric for variable type, $MVT$ value is 0.3. Among these values only one metric value is 0.5 equal to the threshold value and the remaining four values are less than the threshold value. So, according to the above said criterion Label is not a design variable whereas it is a non-design variable.

For the variable Font, interaction metric of variable with methods inside the module, $IM_i$ value is 0.5, interaction metric of variable with methods outside the module, $IM_o$ value is 0, interaction metric of variable with other variables inside the module, $IV_i$ value is 0.2, interaction metric of variable with other variables outside the module, $IV_o$ value is 0, metric for variable type, $MVT$ value is 0.6. Among these values one metric value is 0.5 equal to the threshold value, one metric value is 0.6 greater than the threshold value and the remaining three values are less than the threshold value. So, according to the above said criterion metrics whose values are greater than or equal to threshold value 0.5 are less than fifty percent. Hence, Font is not a design variable whereas it is a non-design variable.

For the variable Font_metrics, interaction metric of variable with methods inside the module, $IM_i$ value is 0.5, interaction metric of variable with methods outside the module, $IM_o$ value is 0, interaction metric of variable with other variables inside the module, $IV_i$ value is 0.6, interaction metric of variable with other variables outside the module, $IV_o$ value is 0, metric for variable type, $MVT$ value is 0.6. Among these values one metric value is 0.5 equal to the threshold value, value of two metrics is 0.6 greater than the threshold value and the remaining two values are 0, less than the threshold value. So, according to the above said criterion metrics whose values are greater than or equal to
threshold value 0.5 are greater than fifty percent. Hence, Font_metrics is a design variable not a non-design variable.

For the variable Color, interaction metric of variable with methods inside the module, $IM_i$ value is 0.5, interaction metric of variable with methods outside the module, $IM_o$ value is 0, interaction metric of variable with other variables inside the module, $IV_i$ value is 0.2, interaction metric of variable with other variables outside the module, $IV_o$ value is 0.3, metric for variable type, $MVT$ value is 0.6. Among these values one metric value is 0.5 equal to the threshold value, one metric value is 0.6 greater than the threshold value and the remaining three values are less than the threshold value. So, according to the above said criterion metrics whose values are greater than or equal to threshold value 0.5 are less than fifty percent. Hence, Color is not a design variable whereas it is a non-design variable.

For the variable Height_offset, interaction metric of variable with methods inside the module, $IM_i$ value is 1, interaction metric of variable with methods outside the module, $IM_o$ value is 0, interaction metric of variable with other variables inside the module, $IV_i$ value is 0.4, interaction metric of variable with other variables outside the module, $IV_o$ value is 0.3, metric for variable type, $MVT$ value is 0.1. Among these values one metric value is 1 greater than the threshold value and the remaining four values are less than the threshold value. So, according to the above said criterion metrics whose values are greater than or equal to threshold value 0.5 are less than fifty percent. Hence, Height_offset is not a design variable whereas it is a non-design variable.

On summarizing the metric values from table 5.2, it is observed that only Font_metrics variable is categorized as design variable and others are categorized as non-design variables.

On similar lines, RFV_Display_Element is considered, metrics are calculated and values are tabulated in table 5.3.
Table 5.3: Values of Metrics for RFV_Display_Element

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Single_element</th>
<th>Flexible_space</th>
<th>Width</th>
<th>Height</th>
<th>X_offset</th>
<th>Y_offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>$IM_i$</td>
<td>0.125</td>
<td>0.125</td>
<td>0.250</td>
<td>0.250</td>
<td>0.250</td>
<td>0.250</td>
</tr>
<tr>
<td>$IM_o$</td>
<td>0</td>
<td>0.009</td>
<td>0.082</td>
<td>0.073</td>
<td>0.039</td>
<td>0.039</td>
</tr>
<tr>
<td>$IV_i$</td>
<td>0</td>
<td>0</td>
<td>0.167</td>
<td>0.167</td>
<td>0.167</td>
<td>0.167</td>
</tr>
<tr>
<td>$IV_o$</td>
<td>0</td>
<td>0</td>
<td>0.047</td>
<td>0.047</td>
<td>0.018</td>
<td>0.018</td>
</tr>
<tr>
<td>$MVT$</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
</tbody>
</table>

RFV_Display_Element has six variables namely Single_element, Flexible_space, Width, Height, X_offset and Y_offset. Values of all the five proposed metrics are calculated for every variable and they are recorded in table 5.3.

For the variable Single_element, interaction metric of variable with methods inside the module, $IM_i$ value is 0.125, interaction metric of variable with methods outside the module, $IM_o$ value is 0, interaction metric of variable with other variables inside the module, $IV_i$ value is 0, interaction metric of variable with other variables outside the module, $IV_o$ value is 0, metric for variable type, $MVT$ value is 0. Among these values only one metric value is 0.125 and the remaining four values are all zeros, less than the threshold value. So, according to the above said criterion Single_element is not a design variable whereas it is a non-design variable.

For the variable Flexible_space, interaction metric of variable with methods inside the module, $IM_i$ value is 0.125, interaction metric of variable with methods outside the module, $IM_o$ value is 0.009, interaction metric of variable with other variables inside the module, $IV_i$ value is 0, interaction metric of variable with other variables outside the module, $IV_o$ value is 0, metric for variable type, $MVT$ value is 0. Among these values all are less than the threshold value. So, according to the above said criterion metrics whose
values are greater than or equal to threshold value 0.5 are less than fifty percent. Hence, Flexible_space is not a design variable whereas it is a non-design variable.

For the variable Width, interaction metric of variable with methods inside the module, $IM_i$ value is 0.250, interaction metric of variable with methods outside the module, $IM_o$ value is 0.082, interaction metric of variable with other variables inside the module, $IV_i$ value is 0.167, interaction metric of variable with other variables outside the module, $IV_o$ value is 0.047, metric for variable type, $MVT$ value is 0.1. Among these values all are less than the threshold value. So, according to the above said criterion metrics whose values are greater than or equal to threshold value 0.5 are less than fifty percent. Hence, Width is not a design variable whereas it is a non-design variable.

For the variable Height, interaction metric of variable with methods inside the module, $IM_i$ value is 0.250, interaction metric of variable with methods outside the module, $IM_o$ value is 0.073, interaction metric of variable with other variables inside the module, $IV_i$ value is 0.167, interaction metric of variable with other variables outside the module, $IV_o$ value is 0.047, metric for variable type, $MVT$ value is 0.1. Among these values all are less than the threshold value. So, according to the above said criterion metrics whose values are greater than or equal to threshold value 0.5 are less than fifty percent. Hence, Height is not a design variable whereas it is a non-design variable.

For the variable X_offset, interaction metric of variable with methods inside the module, $IM_i$ value is 0.250, interaction metric of variable with methods outside the module, $IM_o$ value is 0.039, interaction metric of variable with other variables inside the module, $IV_i$ value is 0.167, interaction metric of variable with other variables outside the module, $IV_o$ value is 0.018, metric for variable type, $MVT$ value is 0.1. Among these values all are less than the threshold value. So, according to the above said criterion metrics whose values are greater than or equal to threshold value 0.5 are less than fifty percent. Hence, X_offset is not a design variable whereas it is a non-design variable.

For the variable Y_offset, interaction metric of variable with methods inside the module, $IM_i$ value is 0.250, interaction metric of variable with methods outside the
module, $IM_o$ value is 0.039, interaction metric of variable with other variables inside the module, $IV_i$ value is 0.167, interaction metric of variable with other variables outside the module, $IV_o$ value is 0.018, metric for variable type, $MVT$ value is 0.1. Among these values all are less than the threshold value. So, according to the above said criterion metrics whose values are greater than or equal to threshold value 0.5 are less than fifty percent. Hence, $Y_{offset}$ is not a design variable whereas it is a non-design variable.

On summarizing the metric values from table 5.3 for RFV_Display_Element, it is observed that not even a single variable among the six variables is a design variable. So, all the six variables come into the category of non-design variables but not the design variables.

Similarly, metric values have been calculated for entire RFV but for the purpose of space constraint only RFV_Text_Line and RFV_Display_Element are explained.

5.5.2 JLex Case Study

Java lexical analyzer, JLex is software system which has been developed in Java. In case of JLex, metrics value for all the variables are calculated in the following manner.

- AspectJ compiler is used to parse the source code and parse tree is constructed.
- Facts-base generator is used to traverse the parse tree and Prolog facts are generated.
- After this, Prolog rules are written and fired against the generated facts.

In above procedure, a separate Prolog rule is written for each metric and values are obtained. In this case study Jlex.CSpec, JLex.Error and JLex.CLexGen classes are considered in which only one variable is added in this version. In Jlex.CSpec class variable $m-public$ is added and after verifying the values obtained for the metrics this variable is categorized as non-design variable. Same is the case with other two classes. One variable that is added in them are also found to be non-design variables.
The proposed metrics suit when applied on two different software systems, namely RFV and JLEX satisfactory results are acquired and hence, validating the proposed metrics empirically.

SUMMARY

This chapter of the thesis discussed about metrics and importance of the metrics. Various criteria for evaluating metrics have been elaborately presented. A detailed discussion about Weyuker’s properties has been provided. The proposed metrics have been assessed against the Weyuker’s properties to demonstrate the effectiveness of the metrics towards their intended purposes. Two software systems RFV and JLEX are considered as case studies, categorization of variables has been demonstrated clearly with the help of these software systems. So, in this manner proposed metrics are validated both analytically using Weyuker’s properties and empirically using RFV and JLEX.