## CHAPTER - 4

### VARIABLE CATEGORIZATION

<table>
<thead>
<tr>
<th>S. No.</th>
<th>Contents</th>
<th>Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>68</td>
</tr>
<tr>
<td>4.2</td>
<td>Concept of Design And Non-Design</td>
<td>70</td>
</tr>
<tr>
<td>4.3</td>
<td>Categorization of Variables</td>
<td>74</td>
</tr>
<tr>
<td>4.4</td>
<td>Design Equivalents</td>
<td>74</td>
</tr>
<tr>
<td>4.5</td>
<td>Metrics Suite</td>
<td>76</td>
</tr>
<tr>
<td>4.6</td>
<td>Metrics for Variable Categorization</td>
<td>77</td>
</tr>
<tr>
<td></td>
<td>4.6.1 Interaction Metric of variable with methods inside the module ( IM_i )</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td>4.6.2 Interaction Metric of variable with methods outside the module ( IM_0 )</td>
<td>79</td>
</tr>
<tr>
<td></td>
<td>4.6.3 Interaction metric of Variable with other variables inside the module ( IV_i )</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>4.6.4 Interaction Metric of Variable with other variables outside the module ( IV_o )</td>
<td>81</td>
</tr>
<tr>
<td></td>
<td>4.6.5. Metric for Variable Type ( MVT )</td>
<td>82</td>
</tr>
<tr>
<td></td>
<td>Summary</td>
<td>83</td>
</tr>
</tbody>
</table>
CHAPTER - 4

VARIABLE CATEGORIZATION

Ignoring unused variables has been discussed in previous chapter. The outcome of this technique shows improvement in the speed and performance of dynamic invariant detection tools by reducing the number of variables based on which invariants are deduced. In order to further improve these factors and in turn the cost, as a next part of methodology to support software evolution, inferred relevant invariants are to be categorized. As a rationale for this purpose variable categorization is considered. In the process of fulfilling this task this chapter discusses about the relation among artifacts, variables, metrics, concept of design and non-design is explained. It also discusses about design equivalents and need for categorizing variables. A set of metrics have been proposed, defined and explained with example. Finally it concludes by elucidating computation of metric values and categorizing variables.

4.1 INTRODUCTION

A software system can be considered or represented at various levels of granularity. At coarse level a software system is considered in terms of subsystems or modules. At the medium level each module consists of data and computations or operations on that data in terms of functions or routines or procedures. At the finest level of granularity it is the data that plays the role in terms of constants and to the maximum extent as variables. All these entities at various levels are considered as artifacts. An artifact can be defined as any interested logical entity [116]. Artifacts are mapped to the physical entities in various ways, depending upon the responsibility taken up by an artifact. Artifacts of various types and granularities that are interrelated can be observed in a software system [116].

Engineering applications require management of several complex and evolutionary components in engineering artifacts. Cars, Ships and Aircrafts are some of the examples of these artifacts.
Continuous evolution of each and every component occurs initially in the design phase of the software development process. Afterwards this can be observed in other phases also, especially in the maintenance phase. Starting from small changes huge changes are possible. During these changes it is very important to consider the meaning of the component and needs to be taken care about.

At the finest level of the component, a variable plays a key role as it is learnt from the earlier chapters and also it has great influence over invariants. With the intention of helping invariant analysis regarding software evolution the idea of variable categorization has been ignited. In literature there exist two types of attribute categorization models. They are semantic based categorization models and non-semantic based categorization models. Among these two models, semantic based categorization model has been proved to be the best one. But, in both these models, designer made the categorization intuitively or conceptually. So, a quantitative model is needed to perform variable categorization.

Software engineering products such as source code, design etc.; various phases of software engineering processes like analysis, design, coding, testing etc.; efficiency of professionals such as productivity of the designer etc., can be measured using software metrics. When software engineering metrics are used in an appropriate manner the success level or failure level of a process, product or person can be quantitatively defined. These are very much helpful in making significant and constructive technical and managerial verdicts that are associated with time, effort, quality, cost etc. Hence, the role of software metrics is highly valuable towards producing qualitative software systems. Software metrics are categorized into two types, they are static software metrics and dynamic software metrics. The metrics which are obtained by analyzing the software statically are known as static software metrics and the metrics which are computed based on gathered data while executing the software are known as dynamic software metrics.
4.2 CONCEPT OF DESIGN AND NON-DESIGN

Software systems evolve very often. In applications of engineering design, engineering artifacts, for example ships, cars and aircrafts, are evolutionary in nature and complex to manage. These artifacts are constituted by number of components that characterize them. Every task in software system development is characterized by some techniques, thumb rules and constraints. These are used by corresponding software engineer in the required phase of software system development, for instance, a designer while designing different components of the artifact. Here, design reuse takes place because, quite often the designer will make use of earlier design’s knowledge in solving the current problem. So, during software evolution to meet constraints and further requirements often existing designs will be modified in routine designing.

Evolution do happen with the intention of improvement of the end product, and this may be regarding additional facilities, better performance, meeting various requirements. So, as a result of evolution many versions will be found for a product in any industry. For example, when Boeing commercial aircraft industry is considered number of versions do exist for Boeing commercial aircraft such as 737 series, 747 series, 757 series etc., as shown in following figure 4.1.

Various evolutions exists for every series as per the customer requirements. For example, Boeing 747-200C is the convertible passenger – cargo variant where as Boeing 747-200F is its freighter variant. Each series has been evolved from an existing series after incorporating proper modifications with the latest technology advancements. Compare to B737-200 series some of the design characteristics of B737-300[217] are illustrated in figure 4.2. It is observed that the wing version that is used in 737-200 has been considered by the designer and it has been customized to meet fresh requirements such as wing tip has been extended and aerofoil has been modified.
A complex artifact may consist of many lower level artifacts. For example, an aircraft consists of wings, landing gear, fuselage, control surface, power plant etc. These artifacts are once again composed of lower level artifacts such as fuselage consists of 737–300 series (Derived from 200) – Some design features

(i) Wing tip extended
(ii) Fuselage stretched
(iii) New flap sections
(iv) Wing aerofoil altered
(v) Additional lateral control spoilers

Figure 4.2: Design Characteristics of B737-300

Figure 4.1: Evolutions in Boeing commercial aircraft
bulkhead, tailcone, frames, nosecone etc., where as wings consists of stringers, ribs, aerofoil sections etc.[218]. Whenever evolution occurs what are the artifacts that can be used as it is they will be used and wherever modifications are required those artifacts are customized as per the new requirements. While doing so, some artifacts are found to be interdependent on others. For instance, wing and tailplane, when change occurs to wing then it requires modification in tailplane. Here, change to wing is affecting tailplane.

Any artifact will be having some distinguishing attributes with it which will be denoted by variables at basic level. These important attributes will be playing a key role in the design of the artifact. For instance, when an element steel is considered, many properties will be associated with that as a stressed element, such as yield stress, thermal conductivity, Poisson’s ratio, melting point and Young’s modulus. Based on the design characteristics only some among all its properties are important in the design perspective. For example, when steel is considered to design a thermally stressed element, thermal conductivity and ductility are the key attributes to the design as shown in the figure 4.3. Whenever there is a change in the values of these key attributes or variables, a new design emerges. Similarly, with respect to elastically stressed element Young’s modulus and yield stress are the key attributes or variables. So, whenever there is a change in the values of these variables it results in a new design. Here, these variables are considered as design variables as they affect the design of an artifact and remaining variables or attributes are considered as non-design variables because, change in these variables will not result in a new version.

Furthermore, when there is a change in an artifact, other dependent artifacts also have to be altered which are affected by this change due to interdependence. For example, when there is a change in the artifact wing in an aircraft, then this change affects other attribute called tailplane which have to be altered.
Properties of the element (steel)

Yield stress
Thermal conductivity
Poisson’s ratio,
Melting point
Young’s modulus
Fatigue
Ductility
Hardness
Weld ability
Erosion property

Influencing variables
of the design

Thermal conductivity
Ductility

Young’s modulus
yield stress

Thermally stressed
Elastically stressed

Figure 4.3: Key variables
4.3 CATEGORIZATION OF VARIABLES

From the above discussion variables can be categorized into two types as design variables and non-design variables. Whenever change occurs to a design variable then it leads to the new version of the artifact. Design variable of an artifact can be a distinguishing property of the artifact or its lower level artifact in the case of composite hierarchy. For example, in military aircrafts, speed, combat radius, rate of climb, weapons system etc., are the design variables. So, these design variables are important attributes of military aircrafts. Depending upon these variables versions of military aircrafts are characterized. Further, some lower level artifacts can be recognized as key characteristics of higher level artifact. For instance, when aircrafts composite hierarchy is considered, in fuselage artifact, bulk head is identified as the key variable because it takes majority of the structural load and influences the design of fuselage. So, bulk head is the design variable of fuselage. Whenever a change occurs to bulkhead it leads to the new version of fuselage whereas changes to non-design variables such as windows and doors will not result a new version.

4.4 DESIGN EQUIVALENTS

When a composite artifact X is considered as depicted in figure4.4 it consists of three variables i, j, k; where ‘i’ is a design variable and j, k are non-design variables. When there is a change in the value of i, it will result in a new version of the artifact, X1. But, when there is a change in j’s value or k’s value, it will not result a new version because j, k are not design variables, they are non-design variables. So, whenever there is a change in non-design variables then it creates a Design Equivalent of the software system or an artifact, but not its new version. So, design equivalents vary only in the value of non-design variables. In figure4.4, X and X1 are two design versions of the software. All the design equivalents of X are captured in a horizontal plane at X and X1’s design equivalents are captured at X1’s horizontal plane.
The variables that rule the design of an artifact are called design variables. The variables that do not rule the design of an artifact are called non-design variables. During software evolution to know the effect of a change that occur categorization of variables is very much helpful. Because, changes in design variables lead to new design version. Therefore, classifying the variables into design and non-design is of utmost importance. In the previous chapter by ignoring unused variables only used, relevant variables are considered to infer invariants. With the intension of applying the concept of design and non-design over those relevant variables and to categorize variables as design variables and non-design variables as shown in figure 4.5, a metrics suit has been developed in this chapter. There are several metrics available in the literature. However, these are not suitable for variable categorization as their purpose is different. To the best knowledge of authors, no metric is found in the literature to categorize dynamic variables that are observed over the dynamic traces during runtime. So, there is an utmost necessity of

Figure 4.4: Design and non-design variables and design equivalents
proposing the metrics suit for the purpose of categorizing the relevant variables. The following section describes the proposed metrics for variable categorization.

Figure 4.5: Architecture to ignore unused variables and variable categorization

The metrics suite for the purpose of variable categorization is presented here under.

4.5 METRICS SUITE

Set of metrics are developed for variable categorization, based on design complexities of variables. These metrics are developed based on the interactions of the variables with the variables, methods within the module and outside the module, type of the variable. A variable is considered as more complex i.e., design variable, if it is larger in size, and has more interactions within and outside a module and need to be given utmost priority. Its properties must be carefully checked and are to be maintained as they are. Because,
whenever there is a change in design variable of any artifact then there is a need to change the design of an artifact and same is to be propagated to related artifacts in the system. This propagation results in either design change or equivalent change in related artifacts based on the amount of change that has happened to that artifact [219,116].

If the interactions of a variable are less, then it is less complex, also referred as non-design variable. When there is a change in non-design attribute of an artifact this leads to its equivalent design change. Whenever this equivalent change is propagated to related artifacts in the system it will result in either equivalent change or no change based on the amount of change that has happened to that artifact. That means equivalents can be replaced with one another without affecting the integrity of the software systems [219,116]. So, they may be given the priority next to that of design variables. In this regard a set of metrics are proposed to categorize the variables.

### 4.6 METRICS FOR VARIABLE CATEGORIZATION

Proposed metrics are developed based on the design complexities of the variables at module level on dynamic traces.

**Table 4.1: Size of variable types**

<table>
<thead>
<tr>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>0</td>
</tr>
<tr>
<td>Character or Integer</td>
<td>1</td>
</tr>
<tr>
<td>Real, Float</td>
<td>2</td>
</tr>
<tr>
<td>Array</td>
<td>3</td>
</tr>
<tr>
<td>Pointer</td>
<td>5</td>
</tr>
<tr>
<td>Record, Struct, Object</td>
<td>6</td>
</tr>
<tr>
<td>File</td>
<td>10</td>
</tr>
</tbody>
</table>

The size value for different variables is an important factor in developing the metrics. So, they are taken from the table 1 that are suggested in [220, 221, 222]. These values are
given based on the design complexities of various variables by the experienced designers. In this thesis variable and variable instance, method and method instance are used interchangeably while discussing the metrics. The proposed metrics to categorize the variables are discussed below.

4.6.1 Interaction Metric of Variable with Methods Inside the Module (IM_i)

This metric quantifies the interactions inside the module. It is defined as the ratio of the number of method instances inside the module that reference the variable instance under consideration to the total number of possible method instances in a module.

\[
IM_i = \frac{NM_{irv}}{TM}
\]

Where \(NM_{irv}\) is number of methods inside the module that reference the variable and \(TM\) is the total number of methods in the module.

![Module A variables and methods interaction](image)

Figure 4.6: Module A methods interaction with its variables

If all the methods in a module reference the variable, then \(IM_i\) is maximum i.e., 1 and if none of the methods in a module reference the variable the value of \(IM_i\) is minimum i.e., 0. Consider a module A shown in figure 4.6 with 8 variables \(v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\) and 5 methods \(A_{m1}, A_{m2}, A_{m3}, A_{m4}, A_{m5}\). Here \(v_1, v_2, v_3\) are files, \(v_4, v_5\) are float type variables, \(v_6, v_7, v_8\) are integers. The figure illustrates the computation of the metric \(IM_i\). In the figure, variables \(v_7\) and \(v_8\) are referenced by all the five methods. Therefore, the value of \(IM_i\) for the variables \(v_7\) and \(v_8\) is \(5/5 = 1\). The variables \(v_1\) and \(v_6\) are
referenced by only one method and the value of $IM_i$ is $1/5 = 0.2$. Similarly, the values of $IM_i$ for other variables can be computed.

4.6.2 Interaction Metric of Variable with Methods Outside the Module ($IM_o$)

This metric quantifies the interactions outside the module. It is defined as the ratio of the number of method instances outside the module that reference the instance of a particular variable under consideration to the total number of possible method instances in the referenced module.

$$IM_o = \frac{\sum_{i=1}^{N-1} \sum_{j=1}^{M} V_{ij}}{\sum_{i=1}^{N-1} \sum_{j=1}^{M} M_{ij}}$$

Where $V_{ij} = 1$ if the $j^{th}$ method of the $i^{th}$ module references the variable under consideration and $V_{ij} = 0$ otherwise, $M_{ij} = 1$ for every $j^{th}$ method of the $i^{th}$ module, $N$ represents the number of referenced modules including the one that is under consideration and $M$ represents the number of methods for each module.

The values 1 and 0 for $V_{ij}$ and $M_{ij}$ are chosen for convenience and normalization. For explaining the computation of metric $IM_o$ the interactions that are shown in figure 4.7 are considered. In the figure module B and module C are outside related modules for module A. Module B has 4 methods $B_{m1}$, $B_{m2}$, $B_{m3}$, $B_{m4}$ and module C has 3 methods $C_{m1}$, $C_{m2}$, $C_{m3}$. Therefore, the total number of methods outside module A is 7. The variable
v1 is referenced by 5 methods and the IM₀ metric value for variable v₁ is 5/7 = 0.71. In a similar way, IM₀ values for other variables can be computed.

4.6.3 Interaction Metric of Variable with Other Variables Inside the Module (IVᵢ)

This metric quantifies the interactions of variable with other variables inside the module. It is defined as the ratio of the number of variable instances inside the module that reference the variable under consideration to the total number of possible variable instances in a module.

\[ IVᵢ = \frac{NV_{im}}{TV} \]

Where \( NV_{im} \) is number of variable instances inside the module that reference the variable under consideration and \( TV \) is the total number of variable instances in the module.

If all the variable instances in a module reference the variable, then \( IVᵢ \) is maximum i.e., 1 and if none of the variables in a module reference the variable the value of \( IVᵢ \) is minimum i.e., 0. Consider a module A shown in figure 4.8 with 8 variables v₁, v₂, v₃, v₄, v₅, v₆, v₇, v₈. The figure illustrates the computation of the metric IVᵢ. In the figure, variables v₁ is accessed by v₆ and v₈. Therefore, the value of IVᵢ for the variable v₁ is 2/8 = 0.25. The variable v₂ is accessed by only one variable v₅ and the value of IVᵢ for v₂ is 1/8 = 0.125. Similarly, the values of IVᵢ for other variables can be computed.
4.6.4 Interaction Metric of Variable with Other Variables Outside the Module \((IV_o)\)

This metric quantifies the interactions of variable with other variables outside the module. It is defined as the ratio of the number of variable instances outside the module that reference the instance of a particular variable under consideration to the total number of possible variable instances that exists in the referenced module.

\[
IV_o = \frac{\sum_{i=1}^{N-1} \sum_{j=1}^{M} V_{ij}}{\sum_{i=1}^{N-1} \sum_{j=1}^{M} W_{ij}}
\]

Where \(V_{ij} = 1\) if the \(j^{th}\) variable of the \(i^{th}\) module references the variable under consideration and \(V_{ij} = 0\) otherwise, \(W_{ij} = 1\) for every \(j^{th}\) variable of the \(i^{th}\) module, \(N\) represents the number of referenced modules including the one that is under consideration and \(M\) represents the number of methods for each module.

In figure 4.9, module A has 8 variables, module B and module C has 3 variables each. All the 3 variables \(x_1, x_2, x_3\) of module B are accessing \(v_1\) and \(v_2\). So, the value of \(IV_o\) for the variables \(v_1\) and \(v_2\) is \(3/6 = 0.5\). The variable \(v_4\) is accessed by \(x_3\) of module B and \(y_1\) of module C so, the value of \(IV_o\) for variable \(v_4\) is \(2/6 = 0.33\). Similarly, the values of \(IV_o\) for other variables can be computed.
4.6.5. Metric for Variable Type (MVT)

It is defined as the ratio of the size of the type of a variable to the maximum size of the variable. The maximum size is the size of the file type variable (see table 4.1 and taken from [219]). The size of a variable or parameter is a specified constant, specifying the complexity of the variable type. By using this metric it can be measured how complex a variable is. Formally it is defined as

\[ MVT = \frac{S}{M_s} \]

where \( S \) is the size of the variable type and \( M_s \) is the maximum size of the variable type. For example, if the type of the variable \( v \) is an integer, then the type metric \( MVT \) for variable \( v \) is \( 1/10 = 0.1 \). Similarly, the type metric \( MVT \) value for variable \( v_1 \) in the above example module A is \( 10/10 = 1 \).

<table>
<thead>
<tr>
<th></th>
<th>( v_1 )</th>
<th>( v_2 )</th>
<th>( v_3 )</th>
<th>( v_4 )</th>
<th>( v_5 )</th>
<th>( v_6 )</th>
<th>( v_7 )</th>
<th>( v_8 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( IM_i )</td>
<td>0.2</td>
<td>0.4</td>
<td>0</td>
<td>0.6</td>
<td>0</td>
<td>0.2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( IM_0 )</td>
<td>0.71</td>
<td>0.28</td>
<td>0</td>
<td>0.71</td>
<td>0.42</td>
<td>0</td>
<td>0.57</td>
<td>0.42</td>
</tr>
<tr>
<td>( IV_i )</td>
<td>0.25</td>
<td>0.125</td>
<td>0.125</td>
<td>0.25</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( IV_o )</td>
<td>0.5</td>
<td>0.5</td>
<td>0</td>
<td>0.33</td>
<td>0</td>
<td>0.66</td>
<td>0.83</td>
<td>0.33</td>
</tr>
<tr>
<td>( MVT )</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.2</td>
<td>0.2</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Table 4.2: Calculated values for various metrics of variables

All the above mentioned metrics are calculated for examples shown in figures 4.6, 4.7, 4.8, and 4.9 respectively and the values are tabulated in table 4.2. 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 to the threshold value (0.5), then they are categorized as design variables. From table 4.2, attribute \( v_1, v_7 \) are categorized as design variables and others are categorized as non-design variables.
SUMMARY

Introduction part of this chapter discussed about artifact and its importance, the manner in which it is related with variable and metrics. Design and non-design concept is exemplified in detail. This concept is applied over relevant variables by explaining the necessity of categorizing variables. In order to implement these ideas a metrics suit is proposed. All the five metrics have been clearly defined and the method of computing values for these metrics has been explained in detail with the help of examples. Based on these computed metric values, variables are categorized as either design variables or non-design variables.

**********