Chapter – 5

Concurrent Maintainability Evaluation and Design of a Software Component

5.1 Overview

The current chapter is an attempt to formulate simple analytic framework to evaluate *maintainability* characteristic of a component. Utilizing digraph and matrix approach, development of *maintainability* index (I_m) is discussed which will help system developers (architects, integrators, decision makers etc.) to rank and select components from the pool of alternatives. Concept of hypothetical best *maintainability* index (I_{bm}) and hypothetical worst *maintainability* index (I_{wm}) is also defined which will help system developers to identify relative comparison of candidates from hypothetical best *maintainability* index and hypothetical worst *maintainability* index. It will also help designers and developers to improve the component *maintainability* characteristic (by analyzing critical attributed factors) by performing sensitivity analysis. The current chapter is organized in the following manner: Section 5.2 discusses the issues related to *maintainability* of software component and customization of ISO 9126 *maintainability* characteristic for software component. In section 5.3 inter and intra interactions of *maintainability* characteristics and associated attributes are identified and digraph representation of the same is developed. Section 5.4 focuses on the methodological framework by utilizing one-to-one mapping of digraph representation, permanent matrix and *maintainability* characteristic (considering attributes and interactive complexity). In section 5.5 *maintainability* analysis and evaluation techniques are discussed. Section 5.6 focuses on method for developing *maintainability* index. Section 5.7 demonstrates and validates the developed approach considering case study. Finally section 5.8 provides concluding remarks of the chapter.

5.2 Introduction

Software maintenance is one of the critical phases as it consumes 60% to 80% of the total life costs (Lientz and Swanson, 1978; Parikh, 1982; Pigoski, 1997). Due to the involvement of huge costs, some organizations are looking at their maintenance process as an area for achieving competitive age (Moad, 1990). Maintainability is a design attribute of a system and plays significant role during system operation. Maintenance activities can be

categorized as: *corrective, preventive, adaptive and perfective maintenance* (Lientz and Swanson, 1980). Corrective maintenance deals with diagnosis and correction of errors. It is very critical because during performing corrective maintenance, system remains idle. *Adaptive maintenance* deals with the activities that modify software to properly interface with a changing environment (hardware and software). *Preventive maintenance* incorporates activities which change software to improve future maintainability or reliability or which provide a better basis for future enhancements. *Perfective maintenance* deals with adding new capabilities, modifying existing functions and making general extensions. For the rapid software development, software organizations now-a-days prefer the usage of component technology. To develop large and complex software systems, software architecture design stage is considered as an important and necessary step. Quality of systems made of COTS products is affected greatly by their complexity that is influenced directly by the number of components and their interactions in the system (Woit, 1997; Upadhyay et al., 2009). Maintenance of CBSS requires several different activities than normal applications (Grover et al., 2007), such as, customizing component using parameterization concept, upgrading the functionality of black-box components (for which code visibility is not available) replacing older version with new version etc. In relation to component *maintainability* perspective various viewpoints can be considered such as – *vendors view point and integrators view point*. Vendors have to think about the *maintainability* of block of code on the basis of it's (re)usage in different applications. In black-box components code source is not visible then integrators have to consider technologies that will maintain the system. In this case visibility is limited to documentation that describes the component's operation and functionality (Mark and Dean, 2000).

In this chapter ISO 9126 model is chosen to further investigate the *maintainability* characteristic of a software component. The description of *maintainability* sub-characteristics based on state of art literature (McCall et al., 1997; Boehm et al., 1978; ISO/IEC-9126, 1991; Dromey, 1995; Sedigh-Ali et al., 2001a; ISO 9126, 2001; Alvaro et al., 2005a; Simão and Belchior, 2003; Bertoa and Vallecillo, 2002a; Goulao et al., 2002b) is mentioned below:

5.2.1 Customizability

It evaluates the capability of a component to be customized according to the user needs. Due to black-box nature of a component it is not possible for a user (system developer,

integrator etc,) to do modifications on the component. But a certain level of modification in terms of tailoring of a functionality of a component is possible by the inclusion of wrappers (adaptars). Wrapping of a component helps in controlling data input and filtering of output data. As compared to black-box component white box-component is expected to have a greater ability in *customization*.

Following attributes contribute to *customizability* of a component:

- *Parameterization*: It indicates the number of parameters offered for change by the component with the number of provided interface. A component offering very few interfaces and large number of parameters is a candidate for more *customization* but difficult to handle. Contrary to this, component offering very large number of interfaces and very less number of parameters does not offer a high degree of *customizability*. Black-box parameterization is measured by the ratio of *number of parameters available for a change to the total number of interfaces supported (offered) by a component*. This can be mapped to a level of satisfaction (LOS) on a scale of 1-5 to measure the *parameterization* of a component. Here value 5 means very high *parameterization* level and value 1 means very weak *parameterization* level, see Table 5.1.

- *Adaptability*: It indicates the ability of a component to adapt itself to a changing environment at runtime. A level of satisfaction scale can be used to measure the adaptability of a component. Value 5 means component is highly *adaptable* while a component with value 1 means weakly *adaptable*. A component agent can be considered as a candidate for highly *adaptable* component.

- *Change control capability*: It indicates the ability of a component to make user aware of current version of a component. A level of satisfaction scale can be used to measure the awareness level provided by a component, where value 5 means component having very high awareness level while value 1 means component providing very weak awareness level, see Table 5.1.

- *Priority*: It indicates the capability of a component to provide prioritize service which some of its functions assume at runtime (ISO/IEC, 1991). A Boolean scale can be used to identify whether the prioritized services are provided or not. If the value is 1, then the level of satisfaction scale can be used to measure the level of prioritized

services provided by a component. Value 5 means component provides very high level of prioritize services while value 1 means component provides very weak level of prioritize services, see Table 5.1.

5.2.2 Testability

It examines the features of a component that can be tested by supporting test cases, tools or test suites.

Following attributes contribute to *testability* of a component:

- *Start up self test*: It indicates the capability of a component to test itself and environment for operation. It is measured by a Boolean scale of 0-1. If value is 1, then a level of scale can be used to measure level of *start up self test* provided by a component. If value is 5 it means component provides very high level of *start up self test* while if it is 1 then it means component provides very weak level of *start up self test*, see Table 5.1.

S.No.	Description	Scale (LOS)
1	Very Low (VL)	1
2	Low (L)	2
3	Moderate (M)	3
4	High (H)	4
5	Very High (VH)	5

Table 5.1 Level of satisfaction (LOS)

- *Trial version*: It denotes the ability of a component to support trials version in order to facilitate user to perform test for the functionality support by a component. Associated with trial version is its timeline usage scale – *limited* or *full version*, whether a component is a *freeware* (f), *shareware* (s), *commercial* (c), see Table 5.2. Level of satisfaction scale can be used to measure the *trial version* level provided by a component. In this scale value 5 means that the component provides very high *trial*

version level while value 1 means that the component provides very weak *trial version* level, see Table 5.1.

S.No.	Trial version Limited/full	Type {f, s, c}	Scale (1-5)
1	Limited	f	{VL, L, M, H, VH}
2	Limited	s	{VL, L, M, H, VH}
3	Limited	c	{VL, L, M, H, VH}
4	Full	f	{VL, L, M, H, VH}
5	Full	s	{VL, L, M, H, VH}
6	Full	c	{VL, L, M, H, VH}

Table 5.2 Component trial version type and scale

- *Test suite provided*: It indicates the presence or absence of a test suite on a Boolean scale of 0 or 1 to measure some properties such as *performance*. If any *test suite* is available then the description of the same is mentioned and level of satisfaction scale on 1-5 can be used to measure the level of *test suite provided* by a component. Here also value 5 means that the component provides very high *test suite* level while value 1 means that the component provides very weak *test suite* level, see Table 5.1.

- *Test materials*: It denotes the existence of other useful test materials like demos, gray code (some level of visible code), logical and data flow diagrams and test cases. Level of satisfaction scale on 1-5 can be used to measure the level of *test materials* provided by a component, with value 5 meaning that the component provides very high *test materials* level while value 1 indicating that the component provides very weak *test materials* level, see Table 5.1.

5.2.3 Changeability

It indicates the effort needed to *modify* a component at ease as per requirements.

Following attributes contribute to the *changeability* of a component:

- *Upgradeability*: It denotes the ability of a component to be *upgraded* to a new version at ease. If *upgradation* requires manual support or intervention leading to system downtime then this will produce negative impact. Level of satisfaction scale on 1-5 can be used to measure the level of *upgradeability* provided by a component. In this values 5 and 1 indicate very high and very low level of *upgradeability* of a component respectively, see Table 5.1.

- *Debugging*: It denotes the ability of a component to support *debugging* (functional) and it evaluates the efficiency of error messages returned by the component in order to understand the erroneous functionality of a component and finally corrects it. For example, error messages such as "*paper jam*" or "*out of paper*" for a printer monitoring component signifies direct understandable meaning in the context of using printer. *The ratio of descriptive and understandable errors to the total number of errors provided by a component can be used as a measure for debugging capability of a component*. This ratio can be mapped to level of satisfaction scale on 1-5 to measure the *debugging* level of a component. Here also values 5 and 1 indicate very high and very low level of *debugging* of a component respectively see Table 5.1.

- *Backward compatibility*: It indicates whether a new component is compatible (backward) with previous versions or not. If a component is not *backward compatible* then the re-user has to re-write the software system to achieve full functionality in order to accommodate changes. Level of satisfaction scale on 1-5 can be used to measure the level of *backward compatibility* provided by a component. Same values as used above show respective high and low *backward compatibility* level of a component, see Table 5.1.

5.3 Concurrent Maintainability Digraph Modeling

Maintainability sub-characteristics and associated attributes do not exist in isolation. They have inter and intra dependencies/interactions (strong, medium, weak, nil), see Table 5.3. These interactions cannot be ignored as they affect each other (directly or indirectly) and thus are responsible for the overall *maintainability* of a component. For example, there exists interaction between *customizability* and *changeability* sub-characteristics of *maintainability*. Thus a weak level of *customizability* (resulted from associated attributes) affects the level of *changeability* (*upgradeability*) that can be achieved for a given component. This means if a component has weak level of change *control capability* (M_{13}) then it is hard to achieve

upgradeability (M_{31}). Similarly, *testability* affects the *changeability* of a component. Likewise other interactions can be obtained, see Table 5.4. To formulate realistic measure of *maintainability* for a software component, it is necessary to consider inter and intra dependencies/interactions concurrently along with sub-characteristics and associated attributes. Conventional methods do not consider such interactions instead they just rely on techniques which calculate individual sub-characteristics or associated attributes values in isolation. Then later by using weighted (average, mean, median, distance etc.,) method the overall *maintainability* of a software component is calculated (Grover et al., 2007). The state of art literature does not provide models that explicitly consider interactions of sub-characteristics and associated attributes concurrently in a unified manner. The current chapter models this using digraph and matrix approach.

A component *maintainability* digraph (weighted) $D_M = (M_I, m_{ij})$ is a set of nodes $M_I = \{v_1, v_2, \ldots, v_i\}$, where '*i*' is from 1 to *n*, representing sub-characteristics. The digraph D_M represents the concurrent consideration of sub-characteristics and their interactions in the form of interactive complexity (inter-intra) of a component. The edges, m_{ij}*'s*, represent the strength of interaction (see Table 5.3, i.e. weights as strong, medium, weak, nil) from sub-characteristic *i* to sub-characteristic *j*. It is to be noted that the number of nodes in a digraph is equal to the number of sub-characteristics (under consideration). In the current study, nodes are three in number and represent sub-characteristics of maintainability – *customizability, testability and changeability.* Sub-node M_{ik}, corresponds to the k^{th} attribute of i^{th} sub-characteristic, and is located in the node M_i representing the sub-characteristic of a component.

Table 5.4 represents the interactions of sub-characteristics and associated attributes. An equivalent digraph is shown in Figure 5.1, where Figure 5.1 (a) represents detailed level of inter-intra interactions of *maintainability* sub-characteristics while Figure 5.1(b) shows conceptual way of visualizing inter-intra interactions of *maintainability* sub-characteristics. It is to be noted that the digraph modeling approach will help in performing brain storming sessions to understand *maintainability* of a component both at the detailed level and at the conceptual level. Since digraph is used for visualization purpose its further analysis is not possible until it is transformed into some analytical or mathematical form/model. So a one-to-one mapping of the considered *maintainability* digraph, (permanent) matrix and a (permanent) matrix function are developed to enable the end user (maintenance personnel,

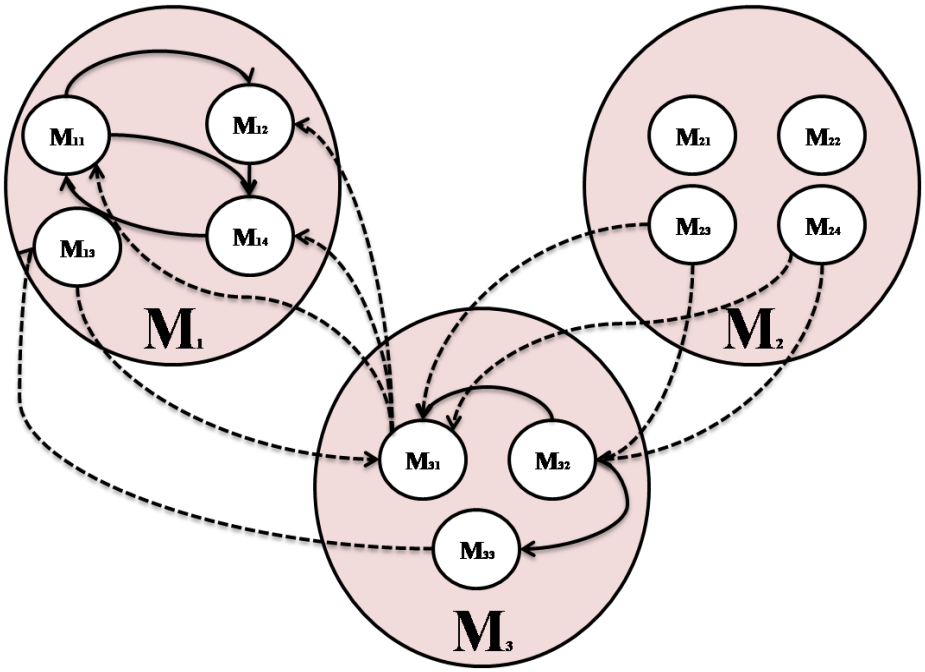
system integrator, acquirer etc.) to identify critical parameters subject to optimized design, development and selection of component from the *maintainability* point of view.

Description	**Strength of Interaction Scale (1 – 5)**
Nil	0
Weak	1
Medium	3
Strong	5

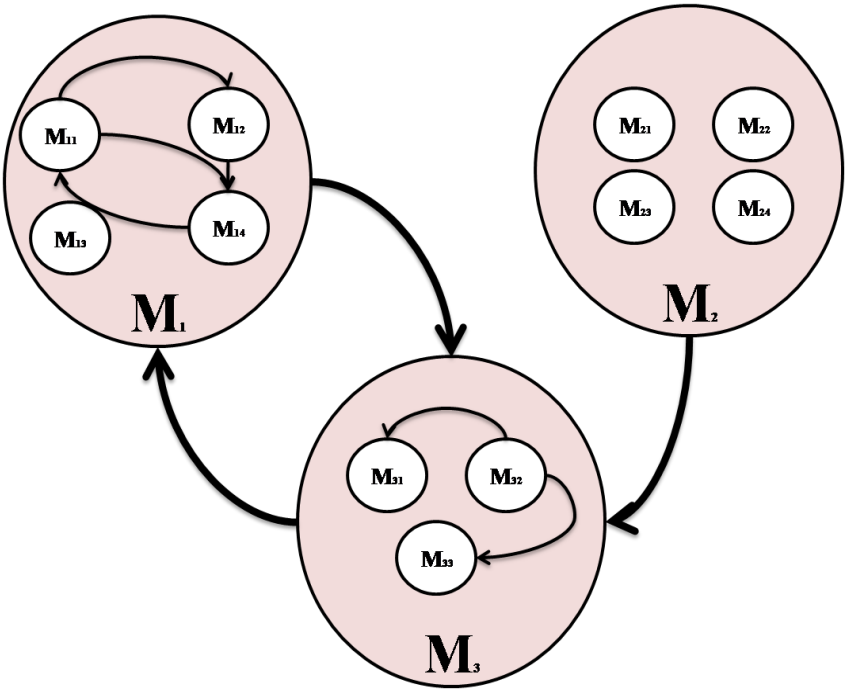
Table 5.3 Strength of interaction (SOI)

Sub-characteristics	**Attributes**	**Interactions**	
		Inter	**Intra**
M_1 *Customizability*	**(M_{11})** Parameterization	----	**M_{12} ; M_{14}**
	(M_{12}) Adaptability	----	**M_{14}**
	(M_{13}) Change control capability	**M_{31}**	----
	(M_{14}) Priority	----	**M_{11}**
M_2 *Testability*	**(M_{21})** Start up self test	----	----
	(M_{22}) Trial version	----	----
	(M_{23}) Test suite	**M_{31} ; M_{32}**	----
	(M_{24}) Test materials	**M_{31} ; M_{32}**	----
M_3 Changeability	**(M_{31})** Upgradeability	**M_{11} ; M_{12} ; M_{14}**	----
	(M_{32}) Debugging	----	**M_{31} ; M_{33}**
	(M_{33}) Backward compatibility	**M_{13}**	----

Table 5.4 Maintainability sub-characteristics interaction (inter and intra level)



(a) Maintainability sub-characteristics interaction digraph at the detailed level



(b) Maintainability sub-characteristics interaction digraph at the conceptual level

Figure 5.1 Maintainability sub-characteristics interaction digraph

The model so discussed helps in achieving *maintainability* (characteristics) expression of a component and is convenient for computer processing. As digraph contains 3 nodes (sub-characteristics), its (permanent) matrix representation is of size 3*3. In the matrix the

diagonal elements represent *maintainability* sub-characteristics of a component while *off-diagonal* elements correspond to interaction of one sub-characteristic on another sub-characteristic of a component (direct or indirect interactions of sub-characteristics). *Permanent* of a matrix is called *variable permanent maintainability function* (*VPF – m*). This expression is the component characteristic of *maintainability* which considers interactive complexity in a concurrent fashion. As the *maintainability* digraph is crucial for the study, it may be prepared by the experts in the area/domain. It is to be noted that two nodes (sub-characteristics) can even have multiple edges which make the graph different from a linear graph and it is known as *multigraph*. Such kind of graph which represents the *maintainability* characteristic of a component is known as component *multigraph* for *maintainability* (*CMG-m*). The *multigraph* in a literature (Deo, 2004; Upadhyay, 2004) is defined as a graph in which edge set *E* is incident on the vertex set *V* such that in the mapping of edge set E and vertex set V, multiple edges between its vertices are allowed. Let, in general, *m* edges incident on the i^{th} and j^{th} vertices, the k^{th} edge is distinguished as $e^k ij$. The multiple edges can be reduced to a single edge if the multiple edges $e^k ij,\ k = 1,2,3...m,$ can be replaced by single edge E_{ij} The sub-graph is represented by $E_{ij.}$

$$E_{ij} = f(e^k ij)\ ,\ k = 1,2,3...m, \tag{5.1}$$

A judicious selection of function $f(e^k ij)$ can give the effect of individual edges and a function of type root mean square value is found to be satisfactory.

5.4 Matrix Representation: Variable Permanent Maintainability Matrix (VPM – m)

In this section, further analysis of maintainability digraph is discussed by developing *maintainability permanent matrix* and *maintainability permanent function*. The *maintainability permanent matrix* provides one-to-one correspondence of the maintainability digraph and thus it retains all information (sub-characteristics and interactions). The permanent matrix (Jurkat and Ryser, 1996) generates a standard function known as *permanent* which is used in combinatorial mathematics. The *permanent* of a matrix is the matrix multinomial and is called *variable permanent maintainability function* (*VPF – m*). This is obtained by taking the determinant of a permanent matrix and changing all the negative signs of the terms of the expression as positive. The *VPF – m* represents and retains all the information of the *maintainability* of the component in a single *maintainability* expression. The interpretation of the same will enhance the *maintainability* of a component

and will also help the evaluators to evaluate component's *maintainability* from the pool of alternatives. Use of this concept in *maintainability* analysis modeling will help in representing interactive complexity as viewed from combinatorial point of view. The benefit of using *VPM – m* and *VPF – m* is that no information about *maintainability* (sub-characteristics and inter-intra interactions complexity) of *component* is lost as expression (*VPF – m*) does not contain any negative sign. Application of this concept will lead to better understanding of *maintainability* analysis of a component.

Let us consider the digraph shown in Figure 5.1 (b) for defining variable permanent *maintainability* matrix for *component*. Let a *diagonal matrix*, M_D, with diagonal elements M_i, $i = 1, 2, 3$ be considered. Here, M_i represents the sub-characteristics of *maintainability*, whose value can be obtained by considering permanent model for associated attributes. Let us also define another matrix, M_O, with *off-diagonal* elements, m_{ij}*'s,* representing the (strength of) interaction between sub-characteristic *i* and *j*. It is to be noted that M_i*'s* and m_{ij}*'s* represent nodes and edges respectively in *maintainability* digraph of the component. The matrix for Figure 5.1 (b) is written as: $VPM - m = [M_D + M_O]$

$$VPM - m = \begin{matrix} 1 & 2 & 3 & \textit{Sub-characteristics} \end{matrix} \\ \begin{bmatrix} M_1 & 0 & m_{13} \\ 0 & M_2 & m_{23} \\ m_{31} & 0 & M_3 \end{bmatrix} \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} \qquad (5.2)$$

The *permanent* of the matrix following standard procedures is written as:

$$VPF - m = [M_1 * M_2 * M_3] + [m_{13} * m_{31} * M_2] \qquad (5.3)$$

The aforementioned permanent matrix and permanent function will provide a completely new way to analyze *maintainability* of a component in a concurrent fashion and is a significant contribution. A systematic use of the approach will lead to the maintainability design and the evaluation of a component and as a whole to CBSS. Expression 5.3 in symbolic form is a useful tool for analysis and consists of various terms as point of combinatorial mathematics. The terms can be arranged in number of groupings. It is to be noted that mathematically each term is a product of three different matrix elements ($N = 3$, in the example). Interpreting same expression it can be noticed that different terms are the set of distinct diagonal elements (M_1, M_2, M_3) and loops of *off-diagonal* elements of different sizes

(m_{ij} m_{ji}, m_{ij} m_{jk} m_{ki}). This generates new meaning to the multinomial from component *maintainability* sub-characteristics interactive point of view, keeping in mind their respective set of attributes and inter-intra interaction among them. By arranging terms of the same structure (set representing sub-characteristics and elements involved in formation of loops) in the same grouping and of different structure in different groupings, *VPF – m* may easily be written as (*N* + 1) groupings, where *N* represents the number of sub-characteristics.

5.5 Maintainability Analysis and Evaluation

On critical analysis of *VPF-m*, expression (5.3), it is inferred that this multinomial contains only distinct sub-characteristics – M_i (*customizability, testability and changeability*), dyads - $m_{ij}^2 / m_{ij} m_{ji}$ and loops – $m_{ij}\, m_{jk}\, \ldots\ldots\, m_{ni}$. In short it can be represented as:

$$
\begin{aligned}
VPF - m &= f(\mathrm{M_i}, m_{ij}^2, m_{ij}m_{jk}m_{ki} \text{ etc}) \quad \{\text{if } m_{ij} = m_{ji}\} \\
&= f\,(\text{Vertices, dyads, loops}) \\
&= f\,(\text{concurrent consideration of sub-characteristics and interactions}) \\
VPF - m &= f'\,(\mathrm{M_i}, m_{ij}m_{ji}, m_{ij}\, m_{jk}\, m_{kl}\, m_{li}, m_{ij}\, m_{jk}\, m_{kl}\, m_{lm}\, m_{mi}) \quad \{\text{if } m_{ij} \neq m_{ji}\} \\
&= f'\,(\text{Vertices, 2-vertex loops, loops}) \\
&= f'\,(\text{concurrent consideration of sub-characteristics and interactions})
\end{aligned}
\qquad (5.4)
$$

The terms of the permanent function *VPF-m* are arranged in (*N*+ 1) groups in the decreasing order of number of vertices M_i (sub-characteristics) present in each term. The last group does not contain any M_i in its terms. It contains only terms such as: m_{ij}^2, $m_{ij}\, m_{jk}\, m_{ki}$, etc. For example, in the aforementioned *maintainability* digraph following groups are considered:

Group 1: The first term (grouping) represents a set of N (N = 3) unconnected component *maintainability* sub-characteristics (*customizability, testability and changeability*) and is written as:

$$/M_1/\ M_2/M_3/$$

A slash represents separation mark between two entities. Analyzing first group means consideration of its each and every entity, i.e. M_1, M_2 and M_3 turn by turn.

The above term can also be written as:

/LOS (M_1)/LOS (M_2)/LOS (M_3)/

Or

/LOS (*customizability*)/LOS (*testability*)/LOS (*changeability*)/

The separation helps in identifying critical parameters and the way to improve them. If the entity 1 i.e. M_1 **(***customizability***)** is highly critical then an in depth study may reveal that this is due to the presence of number of parameters and number of interfaces provided by the component, ratio of number of parameters to number of interfaces, number of services that can be prioritized for a component for its usage. Thus, to improve its LOS the designers and developers of the component have to use standard procedures, 'what-if-analysis' tools, 'cause and effect' approach or some other strategic and analytic measures. Similarly, for other sub-characteristics respective accountable features can be identified. Also, if alternative components are available then by putting respective sub-characteristics LOS value in the term will help in comparing and selecting component as per Group 1.

Group 2: Group is absent as a particular sub-characteristic has no interaction with itself (absence of self-loops).

Group 3: Each term of the third grouping represents a set of two-sub-characteristic loop (i.e. $m_{ij}m_{ji}$) and remaining (N-2) unconnected sub-characteristic. The term is represented as:

$/m_{ij}m_{ji}\ /M_k/$

or

$/\ M'_{ij}\ /M_k/$

For convenience $m_{ij}m_{ji}$ is represented as M'_{ij}. In the above set the entity to be analyzed first is $m_{ij}m_{ji}$. This is a two – sub-characteristics loop and it represents interaction between sub-characteristic i and j. If the resultant value is towards lower side as per the analysis, then an in-depth study is needed to identify contributed factors responsible for its value. Other competitive products can also be studied and compared based upon group factors. Designers of a component can also consider the factors which can increase

component group specific values in order to attain competitive edge. For the present maintainability digraph, the term can be written as:

$$/m_{13}m_{31}/M_2/$$

or

/SOI (m_{13}) SOI (m_{31})/LOS (M_2)/

or

/SOI (from *customizability* to *changeability*) SOI (from *changeability* to *customizability)* /LOS (*testability*)/

In the above expression, the first entity represents interactions between (as per *maintainability* point of view) *customizability* and *changeability*. Depending upon specific domain, strength of interaction can be identified and established. This means how strong or weak *changeability* facilitates and affects *customizability* and vice versa. Later, in totality both the entities should be considered. It is to be noted that the maximum value of both the entities will result into a better contributing factor to the overall *maintainability* characteristic of a component. If interactions are fixed for a specific application then the alternatives for a second entity can be sorted in order to get maximum value in totality.

Group 4: Each term of the fourth grouping represents a set of three-sub-characteristics loops (m_{ij} m_{jk} m_{ki} or its pair m_{kj} m_{ji}) and the composite system measure of the remaining (N-3) unconnected elements. In the current *maintainability* digraph this group has zero entities.

By providing/associating proper physical meaning to the *VPF-m* expression representing interactive complexity concurrently for a component, appropriate interpretation can be obtained. Over all, a general 3-sub-characteristics permanent function will have 3! i.e., 6 entities (sub-sets) arranged in (N + 1) groups. Here, 'N' is the number of sub-characteristics. It is, therefore, possible for *maintainability* analysts and designers to carry out SWOT analysis of their component and take strategic decisions to their advantage.

The diagonal elements of the matrix in equation (5.2) correspond to the three sub-characteristics that constitute component *maintainability* characteristic. The values of these diagonal elements M_1, M_2 and M_3 are calculated as:

$$M_1 = VPF\text{-}m\ (VPM\text{-}m\ (M_1)) \qquad M_2 = VPF\text{-}m\ (VPM\text{-}m\ (M_2))$$
$$M_3 = VPF\text{-}m\ (VPM\text{-}m\ (M_3)) \qquad (5.5)$$

VPM-m(M_1), *VPM-m*(M_2) and *VPM-m*(M_3) are the variable permanent matrices for the three sub-characteristics. The procedure for calculating permanent function for M_1, M_2 and M_3 is the same as for calculating *VPF-m* of equation (5.3).

To get the exact degree of interactions, among sub-characteristics and associated attributes we may have to consider the views of concurrent technical team experts. The final decision on the values may be taken on the recommendations of the concurrent team.

5.6 Component Maintainability Index (I_m)

A *maintainability* index should be able to represent the extent of *maintainability* of a component. The developed *maintainability* index is a quantitative measure of the *maintainability* of a component. This index of component needs to take into account concurrently the value of *maintainability* sub-characteristics for the component and their interactions (among sub-characteristics). *VPF – m* considers interactive complexity of sub-characteristics and associated attributes in a single *maintainability* expression due to this it can be used to generate the measure. Moreover, all terms of *VPF – m* are positive. Thus, increased values of *VPF – m* terms (entities values) will increase the overall value. Based on the I_m the selection and evaluation of the component can be carried out as per *maintainability* point of view. Assigning qualitative or quantitative values to attributes and interactions, evaluation of index can be accomplished. The values can be assigned on a specific scale, see Table 5.1 and Table 5.3. It is to be noted that the strength of interaction among the component *maintainability* sub-characteristics/attributes in fact, depends upon the type of component under analysis. However, the suggested constant values as per Table 5.3 will be useful for the user to compute maintainability of a component. This assessment becomes easier for a concurrent team comprising maintenance personnel, testing personnel and developers rather than an individual.

As *VPF – m* contains various structure invariants of *maintainability* its numerical value becomes powerful means to evaluate component *maintainability*. A computer program in 'C++' language is developed (*Appendix F*) to compute the value of permanent of *maintainability* characteristic. Based on the *maintainability* index various alternatives are evaluated and compared for a system. The best alternative is selected having the highest value of I_m. The hypothetical best *maintainability* index and hypothetical worst *maintainability* index can be obtained by putting maximum and minimum values of diagonal and off diagonal elements of *VPM-m*, based on Table 5.1 and Table 5.3. It is to be noted that for a given domain hypothetical best *maintainability* index and hypothetical worst *maintainability* index can also be obtained by fixing up the interaction values and thereafter putting all diagonal element values to a maximum. Comparison of *maintainability* value of a component (I_m) can be relatively made with the hypothetical *ideal best maintainability* and *worst maintainability indices*. This comparison indicates the level to which the ideal value of *maintainability* of the component can be achieved. This is obtained as:

$$I_{mrb} = \left(\frac{I_m}{I_{bm}} \right) \text{and } I_{mrw} = \left(\frac{I_m}{I_{wm}} \right) \tag{5.6}$$

where, I_{mrb} *and* I_{mrw} are the relative *maintainability* indices, which represent *maintainability* value of the component as "**%**" of the ideal value of the index. This relative index provides *maintenance* personnel and decision maker qualitative information to acquire component. It also gives qualitative information for improving design from *maintainability* point of view to the designers and developers of the component.

To develop maintainability index two approaches can be used-

Bottom up: In this approach the value of sub-characteristics (diagonal elements of equation (5.2)) can be computed by computing the permanent of each individual sub-characteristics considering interactive complexity. Also, off diagonal value (interactions) is computed in the following manner:

- if there exists only one interaction between i^{th} sub-characteristic on j^{th} sub-characteristic then direct value from Table 5.3 is used in place of m_{ij}.
- if i^{th} sub-characteristics have multiple interactions with many attributes of j^{th} sub-characteristics then value of m_{ij} is computed by taking root mean square value of all interactions.

In totality: In this approach lowest level of contributing factors (attributes and inter-intra interactions among each other) for the characteristic have to be put in the form of equation (5.2) and resultant index can be obtained by computing equation (5.3).

5.7 Case Study

A typical component based web-application (Upadhyay et al., 2010; Hong, 2005) is considered to validate and demonstrate the effectiveness of the attempted approach. The problem tackled here is the evaluation and selection of component from a pool of alternatives as per *maintainability* point of view. Initially, six components were identified from the component classified list (Upadhyay and Deshpande, 2010), but based on customer prime requirement two alternatives (C_x and C_y ; ORACLE and IBM DB2 respectively) were short listed. Table 5.5 lists down the *LOS* utilizing Table 4.1 and *SOI* utilizing Table 5.3, of *maintainability* sub-characteristics and associated attributes of component C_x and C_y. The *maintainability* of a complete CBSS depends upon *maintainability* of all components based upon interactions among them. It is very important for an architect and a decision maker to select best component from a pool of alternatives. A spider web diagram, see Figure 5.2, can be utilized for the purpose of ranking. Each attribute of *maintainability* is represented with a spoke.

The hypothetical maximum (best) and hypothetical minimum (worst) value (index) of attributes (from *maintainability* point of view) of a component have to be placed in the respective spoke. Later, each alternative component has to be analyzed for each spoke and a web can be created for visual analysis. It is to be noted that component which is closer to the hypothetical best value will be selected as the prime candidate. It can be seen (Table 5.1 and Table 5.3) that the level of satisfaction for component sub-characteristics and associated attributes are in the range of 1-5 and strength of interaction also ranges between 1 and 5. It is expected that the strength of interactions may vary from one project to another. Thus to get hypothetical best and worst values constant interactions or maximum/minimum interactions values can be considered.

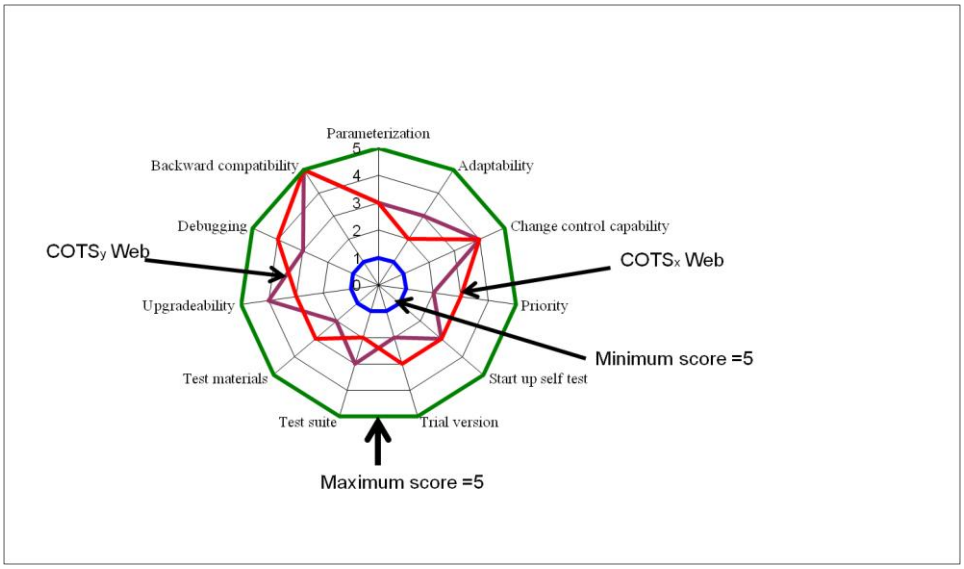


Figure 5.2 Spider diagram for maintainability characteristic

Attributes	LOS C_x	LOS C_y	Interactions M_{ijpq} (from M_{ij} to M_{pq})	SOI *Constant*	SOI		Interactions	SOI *Constant*	SOI	
					Max	Min			Max	Min
M_{11}	3	3	M_{1112}	3	5	1	M_{3114}	3	5	1
M_{12}	3	2	M_{1114}	3	5	1	M_{3231}	4	5	1
M_{13}	4	4	M_{1214}	3	5	1	M_{3233}	3	5	1
M_{14}	2	3	M_{1331}	2	5	1	M_{3313}	2	5	1
M_{21}	3	3	M_{1411}	2	5	1	---	---	---	
M_{22}	2	3	M_{2331}	3	5	1	---	---	---	
M_{23}	3	2	M_{2332}	3	5	1	---	---	---	
M_{24}	2	3	M_{2431}	3	5	1	---	---	---	
M_{31}	4	3	M_{2432}	3	5	1	---	---	---	
M_{32}	3	4	M_{3111}	3	5	1	---	---	---	
M_{33}	5	5	M_{3112}	3	5	1	---	---	---	

Table 5.5 LOS and SOI of components C_x and C_y

Once the strength of interactions is fixed then the permanent index value will be calculated by the variations in the values of attributes level of satisfaction. To get the hypothetical minimum index (value), the level of satisfaction for sub-characteristics and/or associated attributes is to be set to 1. The resultant value after performing permanent computation is the hypothetical minimum index of a component. To get the hypothetical maximum *maintainability* index, the level of satisfaction for associated attributes is to be set to 5. The resultant value after performing permanent computation is the hypothetical maximum *maintainability* index of a component. It is to be noted that decision makers are free to use different scale for different attributes which will result in different hypothetical maximum and minimum *maintainability* indices.

Table 5.5 is used to put values for diagonal and off-diagonal elements. Permanent matrix *VPM-m* for C_x is prepared as:

$$VPM\text{-}m = \begin{array}{c} \\ M_{11} \\ M_{12} \\ M_{13} \\ M_{14} \\ M_{21} \\ M_{22} \\ M_{23} \\ M_{24} \\ M_{31} \\ M_{32} \\ M_{33} \end{array} \begin{array}{c} \begin{array}{cccccccccccc} M_{11} & M_{12} & M_{13} & M_{14} & M_{21} & M_{22} & M_{23} & M_{24} & M_{31} & M_{32} & M_{33} \end{array} \\ \begin{bmatrix} 3 & 3 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 2 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 3 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 3 & 3 & 0 \\ 3 & 3 & 0 & 3 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 3 & 3 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \end{bmatrix} \end{array}$$

The maintainability index for C_x (I_{mx}) is the permanent of the matrix and is given as:

$$I_{mx} = VPF\text{-}m(C_x) = 466560$$

The hypothetical best index of *maintainability* for C_x can be calculated as:

$$M_{11}M_{12}M_{13}M_{14}M_{21}M_{22}M_{23}M_{24}M_{31}M_{32}M_{33}$$

$$VPM\text{-}m = \begin{array}{c} \\ M_{11} \\ M_{12} \\ M_{13} \\ M_{14} \\ M_{21} \\ M_{22} \\ M_{23} \\ M_{24} \\ M_{31} \\ M_{32} \\ M_{33} \end{array} \begin{bmatrix} 5 & 3 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 2 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 3 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 3 & 3 & 0 \\ 3 & 3 & 0 & 3 & 0 & 0 & 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 5 & 3 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \end{bmatrix}$$

$$I_{bm} = 6.757813e{+}07$$

The hypothetical worst index of *maintainability* for C_x can be calculated as:

$$M_{11}M_{12}M_{13}M_{14}M_{21}M_{22}M_{23}M_{24}M_{31}M_{32}M_{33}$$

$$VPM\text{-}m = \begin{array}{c} \\ M_{11} \\ M_{12} \\ M_{13} \\ M_{14} \\ M_{21} \\ M_{22} \\ M_{23} \\ M_{24} \\ M_{31} \\ M_{32} \\ M_{33} \end{array} \begin{bmatrix} 1 & 3 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 3 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 3 & 3 & 0 \\ 3 & 3 & 0 & 3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 1 & 3 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$I_{wm} = 25$$

Similarly, index values for C_y based on Table 5.5 can be computed. Table 5.6 shows the ranking of the components. Decision makers can also set acceptable threshold values of each attributed factors for component and compare the candidates' respective values against

the set threshold values. The final selection of components will depend upon other factor as well such as- business policies, business vision, cost, environmental conditions, legal issues etc.

Component	**Permanent value** I_m	**Hypothetical maximum value** I_{bm}	**Hypothetical minimum value** I_{wm}	**Ranking**
C_x	466560	6.757813e+07	25	II
C_y	622080	6.757813e+07	25	I

Table 5.6 Ranking of components

In the current example, "*In totality*" approach is utilized to calculate *maintainability* index of components C_x and C_y.

Using spider diagram, Figure 5.2, a detailed analysis can be done before selecting the component. For example, if a component is to be used in safety-critical applications then *debugging* attribute is a critical factor. Thus looking at the spoke for *debugging,* decision can be taken for selecting the component which is having high value (close to maximum score). Based on the similar approach designers and developers of the components will also be benefitted. Thus information regarding attribute that is critical for a component to reach to a competitive edge can be obtained.

By performing sensitivity analysis (by varying values) using standard procedures and techniques the index value can be increased to reach to the hypothetical best maintainability index. At the conceptual stage itself they can modify designs and other associated factors to improve the *maintainability* of a component.

5.8 Concluding Remarks

In this chapter, the digraph and matrix approach is utilized to develop various *maintainability* indices of a software component based on attributed factors for *maintainability* characteristic. To achieve this, firstly sub-characteristics and associated attributes and their interactions subject to *maintainability* of a component have been identified then a unique *maintainability* expression is derived from permanent function which is developed from the permanent matrix. This expression yields component's *maintainability* index. The concept of hypothetical best *maintainability* index; hypothetical worst *maintainability* index; and component's relative index from hypothetical best *maintainability* index and hypothetical worst *maintainability* index is also developed. These indices will enable various stakeholders of component oriented project to design, develop, select, acquire and integrate component as per maintainability point of view.

In the next chapter, software failure modes and effects analysis of a software component and component based software system is discussed that can be achieved through graph theoretic approach. Identification of failure index of a software system and/or a software component is also presented. Later, the method for computing *reliability* index of a software system based on heterogeneous architecture styles has also been discussed.