Chapter - 7

Concurrent Quality Modeling and Evaluation of a Software Component

7.1 Overview

In this chapter, the existing quality models (general and component specific) have been reviewed in order to identify component specific characteristics, sub-characteristics, associated attributes and interactive complexity. This chapter also aims to provide methodology based on sound mathematical concepts to concurrently evaluate quality of a software component. The rest of the chapter is structured as follows: In section 7.2, a basic understanding of the current quality concern is established. Section 7.3 describes a new software component specific quality model which overcomes the shortcomings of existing quality models. In section 7.4, methodology based on digraph and matrix approach is developed to evaluate and analyze software component considering interactive complexity. It also deals with the creation of quality index. Section 7.5 validates the component quality model and methodology by performing surveys and also presents illustrative case study. Finally section 7.6 provides concluding remarks of the chapter.

7.2 Introduction

During the last years a significant change is noticed in the paradigm of software development and dissemination. Modern software's are large scale and complex. Many organizations consider implementing such software using software components with the expectations that software components can significantly lower development costs, shorten development life cycle, producing high reliable and high stable product (Tran et al., 1997). Component based software engineering has emerged as a separate discipline which ultimately leads to software system that requires less time to specify, design, test and maintain and yet establish high reliability requirements (Raje et al., 2009; Kallio and Niemela, 2001; Preiss, 2001; Szyperski, 1998). The quality of a component based software system depends upon the quality of individual components and interactions among them considering any rationale or constraint for selecting them (Andreou and Tziakouris, 2007; Upadhyay et al., 2009). A software quality model acts as a framework

for the evaluation of characteristics of the software. It is to be noted that with the development of technology and research, software being used in an increasingly wide variety of application areas and its correct operation is often critical for business success (Rakic and Medvidovic, 2001; Mann et al., 2000). Developing or selecting high quality software component to build complete quality software system is therefore of prime importance. To ensure adequate quality of a software system it is necessary to perform comprehensive specification and evaluation of software component quality. This can be achieved by defining appropriate quality model/standard for the component market that will guarantee component acquirer, high quality components (Gao, 2003; Parminder and Hardeep, 2008). Existing quality models are inappropriate to evaluate quality of software components as they are very general in nature. Very few research studies are available which have contributed to the development of software component specific quality model. More or less each and every model is the customization of ISO-9126 software quality model. To evaluate the quality of end product, a set of quality characteristics that describes the product and forms the basis for the evaluation is required. This set of characteristics and the relationships between them is quality model (Botella et al., 2002). Current techniques and methods concentrates on weighing method (independent) to evaluate quality of software components considering individual characteristics independently without looking at interactions among the quality characteristics. In the current chapter, first component specific quality model is formulated and later evaluative concurrent methodological framework is developed which considers quality characteristics and their interactions at all levels concurrently i.e. in totality in an integrated manner. The evaluation is done on the lines of methodological framework developed in chapter 2.

7.3 Software Component Quality Model (SCQM)

Based on the critical literature survey (Behkamal et al., 2009; IEEE Std 610.12, 1990; ISO 9126-1, 2001; Pressman, 2005; ISO/IEC 14598-1, 1999; Petrasch, 1999; Fizpatrick, 1996; Bohem et al., 1978; Dromey 1995; Khosravi and Guehneuc, 2004; Jacobson et al., 2000; Krutchen , 2000; Grady, 1992; Stefani et al., 2003; Stefani et al., 2004; Bertoa and Vallecillo, 2002a; Alvaro et al., 2005a; Alvaro et al., 2005b;

Raweshdah and Matalkah, 2006; Andreaou and Tziakouris, 2007) a software component quality model (SCQM) has been developed in this chapter which overcomes the shortcomings of existing quality models. In this section description of the *SCQM* is given. Later the validation of the SCQM is discussed in section 7.5. At the highest level the SCQM consists of eight characteristics – *functionality, reliability, usability, efficiency, maintainability, portability, reusability* and *traceability*. A unified measure for attributes is taken on a level of satisfaction scale (LOS) from 1 – 5, see Table 4.1.The description of the SCQM is as follows:

7.3.1 Functionality

This characteristic indicates the ability of a component to provide the required services and functions, when used under the specified conditions. It directly matches with the *functionality* characteristic of ISO 9126 but deals with different set of sub-characteristics. More specifically, *security* and *suitability* sub-characteristic retain their meaning with the only exception that the term *suitability* is replaced with the term *completeness* as it reflects better scope. In addition, *interoperability* is moved under the category of the *reusability* characteristics (not in ISO 9126). Since the components are meant to be reused thus *reusability* is treated as a separate characteristic. Since *accuracy* is considered to be a feature of *reliability* instead of *functionality* it is moved under the category of the *reliability* characteristic (*result set* sub-characteristic). Here, *self-containment* is included which is the intrinsic property of a component. Finally, for the evaluation reason, *compliance* has been removed temporarily because currently there are still no official standards to which component must adhere to.

Following sub-characteristics contribute to the *functionality*:

- ***Self-containment***: *Self-contained* is an intrinsic property of a component and it means component is encapsulated with well-defined interfaces and can be executed independently with minimal outside support (Szysperski et al., 2002).

  Following attributes contribute to *self-containment* of a component

  - *Preconditions and postconditions:* Well defined interfaces have contracts that are described by the presence of *preconditions* and *postconditions* (Szysperski et al., 2002). Boolean scale can be used to identify whether the *precondition* and

postcondition together with which is determined what the component provides and requires, are available or not. If the value is 1 then the level of satisfaction scale can be used to measure the level of precondition and postcondition provided by a component. Value 5 means component provides very high level of precondition and postcondition while value 1 signifies very low level of precondition and postcondition.

- *Modularity*: It indicates the degree of independence of the component, which is the degree of functional cohesion. It can be measured by the ratio of *total number of functions provided by the component itself without external support* to the *total number of functions provided by the component.* High value of ratio means high modularity. Later it can be normalized to the level of satisfaction scale. Value 5 means component provides very high level of *modularity* while value 1 signifies very low level of *modularity*.

- ***Security***: It indicates the ability of a component to control the unauthorized access to the services provided to it. It also deals with whether a security failure of a component will lead to a system wide security failure.

  Following attributes contribute to *security* of a component:

  - *Control access*: It indicates the ability of a component to provide control access mechanisms like *authentication* and *authorization*. Boolean scale can be used to identify whether the *Control access* mechanisms are available or not. If the value is 1 then the level of satisfaction scale can be used to measure the level of *Control access* mechanisms provided by a component. Value 5 means component provides very high level of *Control access* mechanisms while value 1 signifies very low level of *Control access* mechanisms.

  - *Data encryption*: It indicates the ability of a component to provide *data encryption* mechanisms to the secure data it handles. Boolean scale can be used to identify whether the *data encryption* mechanisms are available or not. If the value is 1 then the level of satisfaction scale can be used to measure the level of *data encryption* mechanisms provided by a component. Value 5 means component provides very high level of *data encryption* mechanisms while value 1 signifies very low level of *data encryption* mechanisms.

- *Auditing*: It indicates the ability of a component to keep track of user actions and any unauthorized access. Boolean scale can be used to identify whether the *auditing* mechanisms are available or not. If the value is 1 then the level of satisfaction scale can be used to measure the level of *auditing* mechanisms provided by a component. Value 5 means component provides very high level of auditing mechanisms while value 1 signifies very low level of *auditing* mechanisms.

- *Privilege intensification*: It indicates the ability of a component to identify any flaw in the component which may lead to privilege intensification and thus to system security breach. Boolean scale can be used to identify whether the *privilege intensification* mechanisms are available or not. If the value is 1 then the level of satisfaction scale can be used to measure the level of *privilege intensification* mechanisms provided by a component. Value 5 means component provides very high level of *privilege intensification* mechanisms while value 1 signifies very low level of *privilege intensification* mechanisms.

- ***Completeness***: It indicates the ability of a component to fit into user (re-user) requirements. This provides the overall idea of the level to which component covers the needs of the users in terms of the services offered.

Following attributes contribute to *completeness* of a component:

- *User Satisfaction*: It indicates the ability of a component to meet re-user requirements by offering services as per re-user needs. Here, re-users are not the component providers but are system developers and integrators. It can be measured on a level of satisfaction scale where value 5 means highly satisfied and 1 means weakly satisfied.

- *Service Excitability:* It indicates the ability of a component to provide more than required/expected related service. For example, a component providing a scheduler service might additionally offer group service, a feature that may not be needed originally. It can be measured by the ratio of *total number of functions required by the user* to the *total number of functions provided by the component itself*. High value of ratio means high *service excitability*. Later it can be normalized to the level of satisfaction scale. Value 5 means component provides

very high level of *service excitability* while value 1 signifies very low level of *service excitability*.

7.3.2 Reliability

Reliability is the capability of a component to maintain a specified level of performance when used in stated conditions in a stated period of time. It also indicates the ability of a component to return correct results in a quality manner. This characteristic is directly related to ISO 9126 but with minor modifications in the sub-characteristics notions. *Maturity* is renamed as *service maturity* in order to encompass the meaning of contributed attributes such as – *error prone, error handling, recoverability*, and *availability*. It is to be noted that the sub-characteristics *recoverability* and *fault tolerance* mentioned in ISO 9126 is retained but included in *service maturity* as measurable attributes. Also, a new sub-characteristic *outcome set* is introduced which asserts the correctness and quality of the results returned by the component.

Following sub-characteristics contribute to *reliability* of a component:

- ***Service Maturity****:* It expresses the level of confidence that the component is free from errors. It also indicates the time when the component is available with the services and the ability to recover from failure.

  Following attributes contribute to service maturity of a component

- *Error prone*: It indicates the ability of a component to identify how much it is prone to system errors (complete system failure) and the frequency and the relative importance of those errors. It can be measured by the number of errors occurring per unit of time. Less number of the errors means fewer errors prone to system failure. It can be measured on the level of satisfaction scale where value 5 means highly satisfied (very less number of errors (critical) per unit time) and 1 means weakly satisfied (very large number of errors (critical) per unit time).
- *Error Handling*: It indicates the ability of a component to provide handling mechanisms if any error is encountered. Boolean scale can be used to identify whether the e*rror handling* mechanisms are available or not. If the value is 1 then the level of satisfaction scale can be used to measure the level of Error Handling mechanisms provided by a component. Value 5 means component provides very

high level of e*rror handling* mechanisms while value 1 signifies very low level of Error Handling mechanisms.

- *Recoverability*: It indicates the ability of a component to recover when error occurs (fault tolerance). It also indicates that after recovery if any data or system loss occurs or not. Boolean scale can be used to identify whether the *recoverability* mechanisms are available or not. If the value is 1 then the level of satisfaction scale can be used to measure the level of *Recoverability* mechanisms provided by a component. Value 5 means component provides a very high level of *recoverability* mechanisms while value 1 signifies very low level of *recoverability* mechanisms.
- *Availability*: It indicates the ability of a component to be available for offering services. It is measured on the basis of duration of its availability i.e. the average uptime of the component without serious errors or crashes. It can be measured on the level of satisfaction scale where value 5 means highly satisfied (very high average uptime) and 1 means weakly satisfied (very low average uptime).

- ***Outcome Set***: It indicates the ability of a component to provide correct and quality results. In addition, it indicates that the component support transaction based computation.

  Following attributes contribute to *outcome set* of a component:

- *Correctness*: It indicates the ability of a component to return correct results. In addition, it also indicates the quality of results in terms of computational precision and accuracy. It is measured as a ratio of *'as expected' results* to the *total number of results.* High value of ratio means high *correctness*. Later it can be normalized to the level of satisfaction scale. Value 5 means component provides very high level of *correctness* while value 1 signifies very low level of *correctness*.
- *Transactional*: It indicates the ability of a component to provide transactional processing i.e. rollback facility if transaction fails. Boolean scale can be used to identify whether the *transactional* mechanisms are available or not. If the value is 1 then the level of satisfaction scale can be used to measure the level of *transactional* mechanisms (high/low performance) provided by a component. Value 5 means component provides very high level of *transactional* mechanisms

(high performance) while value 1 signifies very low level of *transactional* mechanisms (low performance).

It is to be noted that reliability can also be assessed by measuring the frequency and severity of failures, the accuracy of output results, the mean time between failures and the ability to recover from failure and the predictability of the program (Raweshdah and Matalkah, 2006).

7.3.3 Usability

The *usability* characteristic in component paradigm has a different meaning due to involvement of different set of users such as system developers as integrators who handle the integration of the components to their systems. Table 7.1 shows summary of *usability* sub-characteristics, associated attributes and measures. For detailed description see chapter 4.

Characteristic	Sub-characteristics	Attributes	Measure
Usability	*Help Mechanisms*	*Help System*	LOS
		Manuals	LOS
		Tutorials and Demos	LOS
		Support Tools and services	LOS
	Learnability	*Time to use*	LOS
		Time to configure	LOS
		Time to administer	LOS
	Operability	*Operation effort*	LOS
		Administration effort	LOS
		Customizability effort	LOS
	Approachability	*Directory listings*	LOS
		Search & Fetch	LOS
		Classification	LOS
		Marketing information	LOS

Table 7.1 Usability characteristic of software component

7.3.4 Efficiency

It is the capability of a component to provide appropriate performance, relative to the amount of resources used under stated conditions. It corresponds to the efficiency characteristics of ISO 9126.

Following sub-characteristics contribute to the *efficiency* of a component:

- ***Time Behavior****:* It indicates the time difference between component's method invocation and getting its response. As a component provides number of methods, an average response time can be considered for the measurement. It is critical for the acceptance or rejection of a component. If a component is having very weak response time then the user would not prefer it for an application even though it provides good functionality.

  Following sub-characteristics contribute to time behavior of a component:

  - *Throughput*: It indicates the ability of a component as to how fast it serves requests and provides results over a given period of time. It is measured as the ratio of *number of successful served requests per unit time*. Later it can be normalized to the level of satisfaction scale. Value 5 means component provides very high level of *throughput* while value 1 signifies very low level of throughput.
  - *Capacity*: In indicates the ability of a component to serve a number of users simultaneously without degrading performance level. It is measured by the number of users who are supported by the component. More the users better the *capacity*. It can be measured on a scale of 1-5 for the level of satisfaction where value 5 means highly satisfied (presence of high capacity) and 1 means weakly satisfied (presence of very less capacity). This is important for web-based components where the number of users is generally large.
  - *Concurrency*: It indicates the ability of a component to provide synchronous or asynchronous invocation. Boolean scale can be used to identify whether the *concurrency* mechanisms are available or not. If the value is 1 then the level of satisfaction scale can be used to measure the level of *concurrency* mechanisms (high/low performance) provided by a component. Value 5 means component provides very high level of *concurrency* mechanisms (high performance) while value 1 signifies very low level of *concurrency* mechanisms (low performance)

- ***Resource Behavior****:* It indicates the ability of a component to utilize resource to run itself.

  Following sub-characteristics contribute to resource behavior of a component:

  - *Memory utilization*: It indicates the amount of memory needed by a component to operate. It is measured by the number of kilobytes of RAM required for its operation. More kilobytes means component utilizes high memory. Later it can be normalized to the level of satisfaction scale. Value 5 means component provides very high level of memory utilization (requires very less KB of RAM) while value 1 signifies very low level of *memory utilization* (requires very large KB of RAM).
  - *Processor utilization*: It indicates the amount of processing time (CPU cycle) needed by a component to operate. It is measured by the average number of CPU cycles required for its operation. More CPU cycles means component utilizes high processing. Later it can be normalized to the level of satisfaction scale. Value 5 means component provides very high level of *processing utilization* (requires very less average number of CPU cycles) while value 1 signifies very low level of *processing utilization* (requires very large average number of CPU cycles).
  - *Disk utilization*: It indicates the amount of disk space needed by a component to operate. It includes both the disk space for installing the component and other supported materials (such as documentation, help files etc.,), as well as the disk space used temporarily during execution time. It is measured by the number of kilobytes of disk required for its operation. More kilobytes means component utilizes high disk space. Later it can be normalized to the level of satisfaction scale. Value 5 means component provides very high level of *disk space utilization* (requires very less disk space) while value 1 signifies very low level of *disk space utilization* (requires very large disk space).

7.3.5 Maintainability

Maintainability is the effort required to replace a software component with the corrected version and to migrate an existing software component from a current component based software system to a new version of the system (Gao et al., 2002). In

chapter 5 a detailed description about *maintainability* characteristic is given. The brief description of *maintainability* sub-characteristics for the proper linking is shown in Table 7.2.

Characteristic	Sub-characteristics	Attributes	Measure
Maintainability	*Customizability*	*Parameterization*	Ratio- LOS
		Adaptability	LOS
		Change control capability	LOS
		Priority	Boolean – LOS
	Testability	*Start up self test*	Boolean – LOS
		Trial version	LOS
		Test suite provided	Boolean – LOS
		Test materials	LOS
	Changeability	*Upgradeability*	LOS
		Debugging	Ratio- LOS
		Backward compatibility	LOS
		Error trace	LOS

Table 7.2 Maintainability characteristic of software component

7.3.6 Portability

This characteristic indicates the ability of a component to be transferred from one environment to another with little or no modification. It corresponds to ISO9126 model in the meaning but two of its sub-characteristics have been transferred to *maintainability* characteristics. *Replaceability* is renamed as *backward compatibility* and has been transferred to the sub-characteristic *changeability* of *maintainability*. Similarly, *adaptability* has been transferred to the sub-characteristic *customizability* of *maintainability*. *Installability* characteristic is retained. In addition to these, *deployability* sub-characteristic is added as a component instance may be deployed only once but installed any number of times (Szysperski et al., 2002).

Following sub-characteristics contribute to portability of a component:

- ***Installability***: It is the capability of a component to be installed on different platforms.

  Following attributes contribute to *installability* of a component:

  - Documentation: It indicates the ability of a component to provide supportive *installable* documentation. Boolean scale can be used to identify whether the *installable* documentation is available or not. If the value is 1 then the level of satisfaction scale can be used to measure the level of *installable* instructions in documentation supported by a component. Value 5 means component provides very high level of *installable* instructions while value 1 signifies very low level of *installable* instructions.
  - Complexity: It is the ability of a component which shows how complex it is to carry out the installation of a component. The level of satisfaction scale on 1-5 can be used to measure the level of *complexity* provided by a component. Value 1 means component provides high *complexity* level while a component with a value 5 means component provides weak *complexity* level (very easily installable).

- ***Deployability***: It is the capability of a component to be deployed on different platforms.

  Following attributes contribute to *deployability* of a component:

  - Documentation: It indicates the ability of a component to provide supportive *delpoyable* documentation. Boolean scale can be used to identify whether the *deployable* documentation is available or not. If the value is 1 then the level of satisfaction scale can be used to measure the level of *deployable* instructions in documentation supported by a component. Value 5 means component provides very high level of *deployable* instructions while value 1 signifies very low level of *deployable* instructions.
  - Complexity: It is the ability of a component which shows how complex it is to carry out the deployment of a component. The level of satisfaction scale on 1-5 can be used to measure the level of *complexity* provided by a component. Value 1 means component provides high *complexity* level while a component with a value 5 means component provides weak *complexity* level (very easily deployable).

7.3.7 Reusability

It is a critical characteristic in the development of component based software system (Jon, 2000), as component can be reused to create more complex applications. Since *reusability* is a driving force for component markets thus it is included as a separate characteristic in SCQM. Reusability improves productivity, maintainability and overall quality of a system.

Following sub-characteristics contribute to the *reusability* of a component:

- ***Interoperability***: It is the ability of a component to be reused. The evaluation of *interoperability* is of prime importance since *reusability* aspect is the cornerstone of CBSS success.

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

  - *Platform Independence*: It indicates the ability of a component to be used in different component models such as CORBA, COM/DCOM and other similar models. It is measured as a ratio of *number of platforms/models supported* to the *number of most used platforms/models.* High value of the ratio indicates high platform independence aspect of a component. This can be mapped to a level of satisfaction on a scale of 1-5. Value 5 means very high platform independence and value 1 means very weak platform independence.
  - *Operating System Independence*: It indicates the ability of a component to be used in different operating systems such as Windows, LINUX, UNIX, AIX etc., and other similar operating systems. It is measured as a ratio of *number of* operating systems *supported* to the *number of most used* operating systems. High value of the ratio indicates high operating system independence aspect of a component. This can be mapped to a level of satisfaction on a scale of 1-5. Value 5 means very high operating system independence and value 1 means very weak operating system independence
  - *Hardware Compatibility*: It indicates the ability of a component to be used in different types of hardware such as personal computers, laptops, PDAs and other similar hardware. It is measured as a ratio of the *number of* hardware *supported* to the *number of most used* hardware. High value of the ratio indicates high

hardware compatibility aspect of a component. This can be mapped to a level of satisfaction on a scale of 1-5. Value 5 means very high hardware compatibility and value 1 means very weak hardware compatibility

- *Data Open-format Compatibility*: It is the ability of a component to output data that is compatible with the well-known formats. Boolean scale can be used to identify whether the *open-format compatibility* is available or not. If the value is 1 then the level of satisfaction scale can be used to measure the level of *open-format compatibility* supported by a component such as XML, ASCII and similar other formats. Value 5 means component provides very high level of *open-format compatibility* while value 1 signifies very low level of *open-format compatibility* i.e. only proprietary format is supported.

- ***Generality***: It is the capability of a component to be generic so that it can be reused in a number of applications.

  Following sub-characteristics contribute to *generality* of a component:

  - *Domain abstraction:* It is the ability of a component to be reused across several domains related to specific functionality that the component offers. Boolean scale can be used to identify whether the *domain abstraction* is available or not. If the value is 1 then the level of satisfaction scale can be used to measure the level of *domain abstraction* supported by a component such as Avionic domain, Safety Critical domain etc. Value 5 means component provides very high level of *domain abstraction* while value 1 signifies very low level of *domain abstraction*.
  - *Reuse maturity:* It is the ability of a component to show/support the reusability record and documentation for future reference. Boolean scale can be used to identify whether the *reuse maturity* is available or not, that means whether component is reused in any domain/application. If the value is 1 then the level of satisfaction scale can be used to measure the level of *reuse maturity* supported by a component. Value 5 means component provides very high level of *reuse maturity* while value 1 signifies very low level of *reuse maturity*.

7.3.8 Traceability

This characteristic expresses the extent of a component's built in capacity of tracking the status of component attributes and component behavior. According to Schmauch (1995) *traceability* refers to the ability to show, at any time, where an item is, its status, and where it has been. It can be noticed that while reconfiguring a component for changed/improved *functionality*, maintainers must perform a full cycle of product evaluation, integration and testing. Replacing an older version of a component with a newer version can also create problems such as security violation, resource usage and performance usage degradation etc. This characteristic is not present in ISO 9126 it is felt that *traceability* of the component is important to *SCQM* as the presence of this will help in validating any violation during modification/replacement of a component. Thus it is treated as a separate characteristic.

Following sub-characteristics contribute to the *traceability* of a component:

- *Behavior Trace* - It is the ability of a component (black box) to track its external behaviors. The major purpose is to track component public visible data or object states, visible events, external accessible functions and the interactions with other components.

  Following attributes contribute to behavior traceability

  - *Performance trace* - It is the ability of a component to record the performance data and to benchmark for each function of a component in a given platform and environment.
  - *State trace* - It tracks the object states or data states in a component.
- *Controllability Trace* - It is the ability of a component that refers to the extent of the control capability to facilitate the customization of its tracking functions.

  Following attributes contribute to controllability trace

  - *Customization trace*: It is the ability of a component to control and set up various tracking functions such as turn-on and turn-off of any tracking functions and selections of trace formats and trace repositories.
  - *Error trace* - It records the error messages generated by a component. The *error trace* supports all error messages, exceptions and related processing information generated by a component.

Each of the above attributes can be measured on the level of satisfaction scale where value 5 means highly satisfied and 1 means weakly satisfied. It is to be noted that separate measure can also be taken depending upon the depth and extensibility of the analysis

The developed component quality model does not discuss *business* characteristics (specific to market) as it does not certify to be quality characteristic specific to a component (in quality context).

7.4 Concurrent Evaluative Methodological Framework

In this section an evaluative methodological framework as an extension to an approach presented in chapter 2 is developed that facilitates concurrent evaluation of quality characteristics. Digraph approach is utilized to model the characteristics in the form of quality digraph for the quality evaluation considering interactions (here relative importance). A permanent matrix which is one-to-one mapping of the quality digraph is developed for an in depth analysis and evaluation of component quality index. It is to be noted that the evaluative methodological framework is equally applicable to any number and level of characteristics under consideration. A concurrent team has to identify quality characteristics for the component.

7.4.1 Quality Digraph

A quality digraph models the component quality characteristics and their interrelationships. This digraph consists of nodes N = $\{n_i\}$ where i = 1 to m {m = 8, as per *SCQM*}. Each node represents the quality characteristic of a component; and a set of edges E = $\{e_{ij}\}$, where e_{ij} is read as an edge from node i to node j. Edges represent the relative importance among the characteristics. The number of nodes is equal to the number of characteristics under consideration for a component based project. A directed edge is shown from node i to node j if node i has relative importance over node j and vice versa. A general quality graph for a component as per *SCQM* is shown in Figure 7.1.

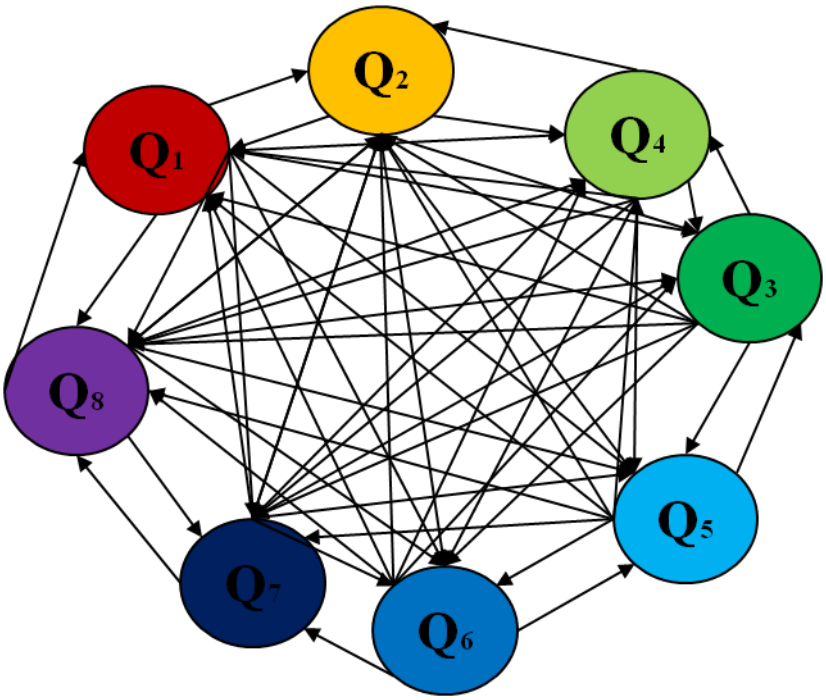

Figure 7.1 Software component quality digraph

The quality digraph gives a graphical representation of the characteristics and their relative importance for quick visual appraisal. In this case visual analysis is expected to be difficult and complex as the digraph is too complex due to the presence of characteristics as nodes and relative importance as edges. To demonstrate the application of the developed methodology, a case study of component quality characteristics for a given component based project is considered (will be discussed later in the case study). Let the component quality characteristics of interest be – *reliability* (Q_1), *usability* (Q_2) and *maintainability* (Q_3). In actual practice, a number of characteristics need to be considered (Figure 7.1). Let us consider following rules to develop quality digraph:

- *Reliability* characteristic is considered more important than *usability* and *maintainability* characteristics.
- *Maintainability* characteristic is considered more important than *usability*.
- *Usability* is not as important as any other characteristics.

Based on above mentioned rules following inferences can be obtained for developing its quality digraph:

- Direct edge exists from Q1 to Q2 and Q3.

- Direst edge exists from Q3 to Q2.
- No direct edge from Q2 to Q1 and Q3.

A digraph G_{RUM} (RUM = Reliability, Usability, Maintainability), considering aforementioned rules and inferences is developed and shown in Figure 7.2.

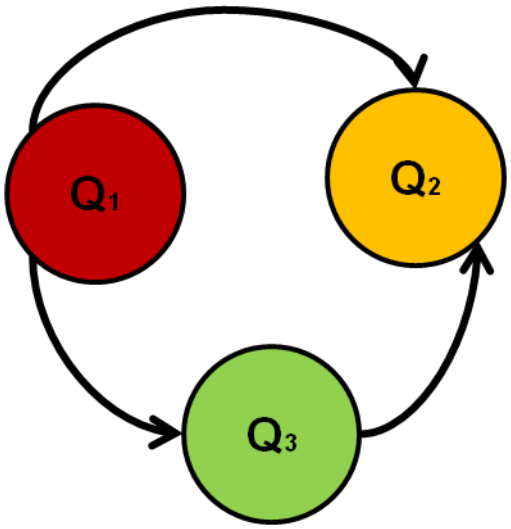

Figure 7.2 $\boldsymbol{G_{RUM}}$ Digraph (***RUM***: Reliability, Usability and Maintainability)

Though digraph (G_{RUM}) seems to be simple, but if G_{RUM} is expanded by considering all its sub-characteristics and associated attributes then its visual analysis will be difficult. To overcome this constraint matrix approach is utilized which provides one-to-one correspondence of digraph.

7.4.2 Matrix Representation of Quality Digraph

Extending matrix models concept presented in chapter 2, a matrix called '*Quality Relative importance matrix*', (QRIM), is defined. This is represented by a binary matrix $A_{RUM} = (q_{ij})$, where q_{ij} represents the relative importance between characteristics i and j such that

$q_{ij} = 1$, if the characteristic 'i' has relative importance over characteristic 'j' and

$= 0$, otherwise.

It is to be noted that $q_{ii} = 0$ for i, as a characteristics cannot have relative importance over itself. The QRIM for G_{RUM} is written as:

$$A_{RUM} = \begin{matrix} Q_1 & Q_2 & Q_3 \text{ Characteristics} \\ \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} & \begin{matrix} Q_1 \\ Q_2 \\ Q_3 \end{matrix} \end{matrix} \qquad (7.1)$$

It can be seen that the above matrix, equation (7.1), consider the presence or the absence of relative importance of characteristics only. No consideration to the value of quality characteristics is given as all the diagonal elements are zero. To incorporate this into account a new matrix called '*Characteristics Quality RIM*', (CQRIM), is defined. The CQRIM for G_{RUM} is written as:

$B_{RUM} = [QI - A_{RUM}]$

$$B_{RUM} = \begin{matrix} Q_1 & Q_2 & Q_3 \text{ Characteristics} \\ \begin{bmatrix} Q & -1 & -1 \\ 0 & Q & 0 \\ 0 & -1 & Q \end{bmatrix} & \begin{matrix} Q_1 \\ Q_2 \\ Q_3 \end{matrix} \end{matrix} \qquad (7.2)$$

where I is an identity matrix and Q is a variable representing the measure of the quality characteristic. This matrix is analogous to characteristics matrix in the graph theory (Deo, 2004). Looking at the matrix in equation (7.2), it is noted that the value of all diagonal elements is identical i.e. the presence or measure of quality characteristic is taken to be the same. In practice this is not true. Also, relative importance value can be any value except 0 or 1. Thus there is a need of a matrix which will consider either presence or measure of quality characteristics and presence or measure of relative importance to cover broad level of evaluation. To suffice the above mentioned statement a new matrix called '*Variable Characteristic Quality Presence and RIM*', (VCQPRIM), is defined. The VCQPRIM for G_{RUM} is written as:

$C_{RUM} = [M_Q - M_O]$

$$C_{RUM} = \begin{array}{c} \begin{matrix} Q_1 & Q_2 & Q_3 \text{ Characteristics} \end{matrix} \\ \begin{bmatrix} Q_1 & -q_{12} & -q_{13} \\ 0 & Q_2 & 0 \\ 0 & -q_{32} & Q_3 \end{bmatrix} \begin{matrix} Q_1 \\ Q_2 \\ Q_3 \end{matrix} \end{array} \tag{7.3}$$

where, M_Q is a diagonal matrix with the diagonal elements representing a variable measure of the i^{th} quality characteristic and M_O represents the off diagonal elements specifying the relative importance between characteristics. If the quality characteristic of a component is very high then a maximum value is assigned. It is to be noted that now either the presence or the measure of quality characteristic and the presence or the measure of relative importance is considered by the matrix. The characteristics polynomial of a matrix equation (7.3) is written by calculating the determinant of a matrix.

$$\text{Det}(C_{RUM}) = Q_1\, Q_2\, Q_3 \tag{7.4}$$

It can be seen that due to the involvement of value in off-diagonal elements most of the information about relative importance is lost, equation (7.4). This happens because matrix only considers the presence of the relative importance from node *i* to node *j* i.e. if characteristic *i* is more important than characteristic *j* then there is a direct edge from characteristic *i* to characteristic *j*. But if characteristic *i* is less important than characteristics *j* then there is no direct edge from characteristic *i* to characteristic *j* and vice versa. In that case q_{ij} (q_{ji}) becomes zero on the matrix representation. This zero causes many terms of the characteristic multinomial to become zero (as there are no relative importance loops in the corresponding digraph). Hence the relative importance between characteristic *i* and characteristics *j* and characteristic *j* and characteristic *i* is distributed on scale 0 to 1, and is defined as:

$$q_{ji} = q_{ij} - 1 \tag{7.5}$$

The modified digraph for G_{RUM} considering relative importance is shown in Figure 7.3.

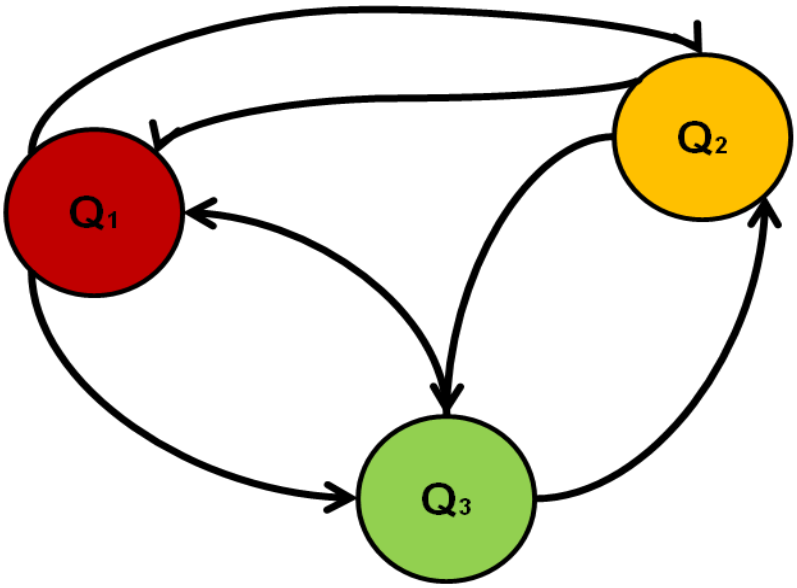

Figure 7.3 Modified digraph for G_{RUM} (complete relative importance)

The matrix of the above digraph is written as:

$$D_{RUM} = \begin{array}{c} \begin{matrix} Q_1 & Q_2 & Q_3 \text{ Characteristics} \end{matrix} \\ \begin{bmatrix} Q_1 & -q_{12} & -q_{13} \\ -q_{21} & Q_2 & -q_{23} \\ -q_{31} & -q_{32} & Q_3 \end{bmatrix} \begin{matrix} Q_1 \\ Q_2 \\ Q_3 \end{matrix} \end{array} \quad (7.6)$$

The determinant of this ($\det(D_{RUM})$) is known as '*Variable Characteristic Quality Multinomial*', (VCQM),and is written:

$$\det(D_{RUM}) = \text{VCQM} = Q_1*Q_2*Q_3 - Q_1*q_{23}*q_{32} - q_{21}*q_{12}*Q_3 - q_{21}*q_{13}*q_{32} - q_{31}*q_{12}*q_{23} - q_{31}*q_{13}*Q_2 \quad (7.7)$$

The VCQM contains terms both of positive and negative signs. It is the comprehensive tool for analysis in symbolic form. While calculating VCQM value for quality analysis, some information about quality characteristic and relative importance measures is lost (after putting their respective measures). This is due to the cancellation of some terms and subtraction operation in the process of computing VCQM. In order to avoid loss of information during quality analysis and evaluation in critical cases, a new

matrix, '*Variable Permanent Quality Matrix'*, (VPQM) is developed. This matrix, E_{RUM}, retains all the multinomial terms with no subtraction operation and hence preserve information about the component's quality characteristic and relative importance measures (Upadhyay et. *al.*, 2009; Jurkat and Ryser, 1996). The determinant of the matrix, (det(E_{RUM})), is known as '*Variable Permanent Quality Function'*, (VPQF). The VPQM is written as:

$E_{RUM} = [M_Q + M_O]$

$$E_{RUM} = \begin{matrix} Q_1 & Q_2 & Q_3 & \text{Characteristic} \\ \end{matrix} \begin{bmatrix} Q_1 & q_{12} & q_{13} \\ q_{21} & Q_2 & q_{23} \\ q_{31} & q_{32} & Q_3 \end{bmatrix} \begin{matrix} Q_1 \\ Q_2 \\ Q_3 \end{matrix} \quad (7.8)$$

The determinant is also known as variable permanent quality multinomial and is written:

$$\begin{aligned} \det(E_{RUM}) = VPQF &= [Q_1 * Q_2 * Q_3] \\ &+ [q_{23} * q_{32} * Q_1 + q_{12} * q_{21} * Q_3 + q_{13} * q_{31} * Q_2] \\ &+ [q_{13} * q_{32} * q_{21} + q_{12} * q_{23} * q_{31}] \end{aligned} \quad (7.9)$$

It can be inferred that the terms present in VCQM and VPQF are the same but they differ in the signs. For critical analysis of component's quality, the VPQF of equation (7.9) is written in a standard form as (N + 1) groups, (where N = 3 characteristics in the current study). The multinomial, i.e., the permanent function when written in (N + 1) groups, presents an exhaustive way of analysis of *component's quality* at the different levels. It helps in identifying different critical quality characteristics and provides an insight to improve them.

A complete permanent function has been written in a systematic manner for unambiguous and unique interpretation. In short it can be represented as:

$$
\begin{aligned}
\text{VPQF} &= f\,(Q_i, \; q_{ij}^2, \; q_{ij}\, q_{jk}\, q_{ki} \text{ etc}) \quad \{ \text{if } q_{ij} = q_{ji} \} \\
&= f\,(\text{Nodes, dyads, loops}) \\
&= f\,(\text{quality elements}) \\
\text{VPQF} &= f'\,(S_i, \; q_{ij} q_{ji} \; , \; q_{ij}\, q_{jk}\, q_{kl}\, q_{li} \; , \; q_{ij}\, q_{jk}\, q_{kl}\, q_{lm}\, q_{mi}\,) \quad \{ \text{if } q_{ij} \neq q_{ji} \} \\
&= f'\,(\text{Nodes, 2-vertex loops, loops}) \\
&= f'\,(\text{quality elements})
\end{aligned}
\qquad (7.10)
$$

The terms of the permanent function, VPQF**,** are arranged in (N+ 1) groups in the decreasing order of number of nodes/characteristics Q_i, where N represents number of quality characteristics. The first group contains terms with N unconnected Q_i*'s***.** Each successive group has one less characteristic than the previous group and rest of the elements are the combination of dyads and loops. The last group does not contain any Q_i in its terms. It contains only terms such as q_{ij}^2, $q_{ij}\, q_{jk}\, q_{ki}$, etc. Following are the group specification for the quality characteristics under consideration:

- Group 1**:** The first term (of grouping) represents a set of N unconnected quality characteristics i.e., Q_1, Q_2, ..., Q_n. For example, if the analysis is carried out for the *RUM* model, the first set is

  /Q_1/Q_2/Q_3/

  Or

  /*Reliability*/*Usability*/*Maintainability*/

  If the entity 2 i.e. Q_2 (Usability - *Learning Mechanisms*) characteristic is more critical then an in depth study may reveal that this attributes to the presence of percentage of functional elements, interfaces, methods and configurable parameters described in manuals, percentage of functional elements incorrectly described in manuals, difference in component version, ratio of words per functional element, ratio of words per interface, ratio of words per methods and ratio of words per configuration parameters, percentage of functional elements correctly used after reading manuals, percentage of functional elements covered in demos and tutorials,

demos and tutorials version difference etc. By the application of appropriate techniques, the attributed factors can be optimized and can be made available for easy use. Using similar approach, the attributed factors of other sub-characteristics can be identified.

- Group 2: Group is absent as a particular characteristic has no interaction/relative importance with itself (absence of self-loops).
- Group 3: Each term of the third grouping represents a set of two-element component quality relative importance loops (i.e.,$q_{ij}\ q_{ji}$)and measure of the remaining (N-2) unconnected elements/characteristics. Group has three terms as follows:

$$/(q_{23}\, q_{32}\, Q_1)/q_{12}\, q_{21}\, Q_3)/(q_{13}q_{31}Q_2)/$$

- Group 4: Each term of the fourth grouping represents three-element component quality relative importance loops (i.e.,$q_{ij}\ q_{jk}\ q_{ki}$) and its pair. This group has two terms as follows:

$$/(q_{13}q_{32}\, q_{21})/(q_{12}q_{23}q_{31})/$$

After identifying these combinatorial terms and by associating a proper physical meaning with them, a new mathematical meaning of this multinomial is obtained. In general, the permanent (*VPQFA*) of an N*N matrix, A with entries $a_{i,j}$ is defined by (Forbert and Marx, 2003) as:

$$VPQFA\ = \sum_{P} \prod_{i=1}^{N} a_i, P(i)\,, \tag{7.11}$$

where, the sum is overall permutations P. In the present work a computer program, PERMANENT, is used (*Appendix F)* for calculating the value of VPQF of a square matrix of dimension N*N, where N is the number of characteristics under study. It is to be noted that the computational effort will increase if $N > 100$.

7.4.3 Component Quality Evaluation (Q_E) and Quality Index (I_Q)

The quality evaluation of a component (Q_E) can be done by evaluating diagonal elements and establishing relative importance of off diagonal elements. The diagonal elements of the matrix in equation (7.8) correspond to the three characteristics that constitute a component quality (under consideration). The values of these diagonal elements Q_1, Q_2, Q_3 are calculated as:

$$Q_1 = VPQF(VPQMQ_1) \quad Q_2 = VPQF(VPQMQ_2) \quad Q_3 = VPQF(VPQMQ_3) \qquad (7.12)$$

where, $VPQMQ_1$, $VPQMQ_2$, and $VPQMQ_3$ are the variable permanent quality matrices for three quality characteristics. The procedure for calculating values of $VPQMQ_1$, $VPQMQ_2$, and $VPQMQ_3$ is the same as for calculating *VPQF* of equation (7.9). For this purpose, the sub-characteristics of component's quality characteristics are considered. By substituting measures of characteristics and their relative importance in *VPQF* an index called '*quality index' I_Q* can be developed. This index can be used to evaluate component alternatives. As *VPQF* contains +ve terms/values of Q_i and q_{ij}, therefore higher values of both results in higher value of *VPQF*. The value of Q_i should preferably be obtained from the available or estimated data. If a quantitative values of the characteristic attributes are not available then a ranked value judgment on a scale of 1 to 5 (LOS), see Table 4.1, can be used. If quantitative values of the characteristic attributes are available then, normalized values of an attribute assigned to the alternate component characteristics are calculated by m_i/m_j, where m_i is the measure of the attribute for the i^{th} alternative and m_j is the measure of attribute for j^{th} alternative. The measure m_j has the higher value of the attribute among the considered alternatives. The ratio is valid for beneficial attributes only, which means high values of these attributes are preferred. The non-beneficial attributes have to be normalized such that their lower values are preferred and is computed by m_j/m_i where m_j is the measure that has lower value of the attribute among the considered alternatives. Once a qualitative attribute value is established on a scale, then the normalized values of the attribute assigned for different alternatives are calculated in the same manner as that of quantitative attributes values.

The relative importance between two attributes (characteristics) for a given component (based on component based project problem) is also assigned a value similar to one described for qualitative attribute and is arranged in 11 classes, see Table 7.3. The relative importance implies that an attribute *i* is compared with attribute *j* in terms of its relative importance for the given problem domain. It may be noted that one may choose any scale for Q_i and q_{ij}, but the final ranking will not change as these are relative values. It is however, desirable to choose a lower scale for Q_i and q_{ij} to obtain a manageability value of the quality index.

Class description	q_{ij}	$q_{ji} = 10 - q_{ij}$	q_{ij}	$q_{ji} = 1 - q_{ij}$
Attribute *'i'* is exceptionally less important than the attribute *'j'*	0	10	0.045	0.955
Attribute *'i'* is extremely less important than the attribute *'j'*	1	9	0.135	0.865
Attribute *'i'* is very less important than the attribute *'j'*	2	8	0.255	0.745
Attribute *'i'* is less important than the attribute *'j'*	3	7	0.335	0.665
Attribute *'i'* is slightly less important than the attribute *'j'*	4	6	0.410	0.590
Attribute *'i'* and *'j'* are equally important	5	5	0.500	0.500
Attribute *'i'* is slightly more important than the attribute *'j'*	6	4	0.590	0.410
Attribute *'i'* is more important than the attribute *'j'*	7	3	0.665	0.335
Attribute *'i'* is highly important than the attribute *'j'*	8	2	0.745	0.255
Attribute *'i'* is extremely more important than the attribute *'j'*	9	1	0.865	0.135
Attribute *'i'* is exceptionally more important over the attribute *'j'*	10	0	0.955	0.045

Table 7.3 Relative importance scale between component quality characteristics

7.5 Validation and Discussion

The aim of this section is to validate and discuss the developed component quality model and evaluation methodology. To do so, different perspectives of quality can be considered, such as acquirer view, integrator view, manager view, developer view, etc. For instance, from system user's point of view *usability* of a software component is relatively more important than any other characteristics. While from maintainers viewpoint *maintainability* is more important than the rest of the characteristics. Thus to measure the overall quality of a product/software component all the concerned view points (stakeholders) have to be considered. It may be noted that practically it is not possible to satisfy all stakeholders' needs at the same time. To get a broad view of the validation, we chose twenty one persons divided into three groups – *Researchers* '*R*', *Academicians* '*A*' and *Developers* '*D*'. The validation was done in two phases: in the first phase the proposed quality model was assessed and validated based upon 21 respondents. The aim of this phase was to get different interpretation and perception of the developed component quality model and to see the usefulness of the developed model. For the second phase, same group were asked to participate in concurrent evaluation of '*RUM*' of a component (*Case study, sub-sub-section 7.5.1*). The validation helped in identifying time and effort required in practice to apply developed methodology by different people. The questionnaires (*Appendix B*) were designed for deriving effectiveness of the quality model. The questionnaires were distributed to groups. A brief workshop was held in order to discuss the meaning of component specific quality characteristics, sub-characteristics, associated attributes and measures. The groups (evaluators) were then asked to fill up the questionnaires.

Table 7.4 and Table 7.5 presents the results of the survey performed with the *researchers*, *academicians* and *practitioners* for phase I. Value of '*y*' indicated relation with quality model and '*n*' indicated no relation with the quality model (i.e. element need to be discarded). In case of disagreement it is displayed with a –1. The satisfaction level and confidence level measure were also associated with the answer to the questions. The satisfaction level and confidence level were marked from 1 to 5, where 1 represented very weak confidence/satisfaction level and 5 showed great confidence/satisfaction level.

Characteristics	Group 'R' R_1 R_2 R_3 R_4 R_5 R_6 R_7							Group 'A' A_1 A_2 A_3 A_4 A_5 A_6 A_7							Group 'D' D_1 D_2 D_3 D_4 D_5 D_6 D_7							Level of Agreement %
Functionality	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	100
Reliability	y	y	y	y	y	y	y	y	y	n	y	n	y	y	y	y	y	y	y	y	y	90.4
Usability	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	100
Efficiency	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	100
Maintainability	y	y	y	y	y	y	y	n	y	y	y	y	y	y	y	y	y	y	y	y	y	95.2
Portability	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	100
Reusability	y	y	y	y	y	y	y	y	n	y	y	y	y	y	y	y	y	y	y	y	y	95.2
Traceability	y	y	y	y	y	-1	y	y	y	n	y	y	y	y	y	y	y	y	y	-1	y	85.2
Level of Agreement %	100	100	100	95.2	100	95.2	100	95.2	95.2	90.2	100	85.6	100	100	95.2	100	95.2	95.2	100	95.2	100	

Table 7.4 Level of agreement for software component quality model

It is worth noting here that Group 'R' and Group 'D' had almost similar viewpoint of the model. A little variation (which is acceptable) is seen in the viewpoint of group 'A', this was because people from these groups were less involved in practical aspects of component domains. The survey result established the fact that the quality model covers an in depth coverage and understanding of software component quality concerns and terminologies. Participants were satisfied with the *software component quality model*. The component quality model survey result has also shown increased level of exposure in understanding new technologies. This will give an exposure to both academia and industry to perceive component products according to their project goals and requirements. A slight disagreement was found in the *traceability* characteristic as the feedback obtained from Group 'R' and Group 'D' indicated that they agreed that though this characteristic should be retained but it should come under *maintainability*. But stronger feedback was received from other participants in support of treating *traceability* as an individual characteristic. Therefore the position of this characteristic was retained. A detailed result regarding participants' opinion about component quality sub-characteristics is shown in Table 7.5. The Table 7.5 indicates some disagreement among the participants. This is because of the fact that some of the evaluators were not fully aware of the nomenclature and notions of the sub-characteristics while other who were well versed in this area wanted to retain the sub-characteristics but at the same time

wanted to transfer them to other sub-characteristics. However, the overall result gave a strong indication of retaining the quality model without any further modification.

Characteristics		Group 'R' R_1 R_2 R_3 R_4 R_5 R_6 R_7	Group 'A' A_1 A_2 A_3 A_4 A_5 A_6 A_7	Group 'D' D_1 D_2 D_3 D_4 D_5 D_6 D_7	Level of Agreement %
Functionality	*Self-containment*	y y y y y y y	y y y y y y y	y y y y y y y	100
	Security	y y y y y y y	y y y y y y y	y y y y y y y	100
	Completeness	y y y y y y y	y y y y y y y	y y y y y y y	100
Reliability	*Service Maturity*	y y y y y y y	y y y y y y y	y y y y y y y	100
	Outcome Set:	y y y y y y y	y y n y n y y	y y y y y y y	90
Usability	*Help Mechanisms*	y y y y y y y	y y y y y y y	y y y y y y y	100
	Learnability	y y y y y y y	y y y y y y y	y y y y y y y	100
	Operability	y y y y y y y	y y y y y y y	y y y y y y y	100
	Approachability	y y y y y y y	y y y y y y y	y y y y y y y	100
Efficiency	*Time Behavior*	y y y y y y y	y y y y y y y	y y y y y y y	100
	Resource Behavior	y y y y y y y	y y y y y y y	y y y y y y y	100
Maintainability	*Customizability*	y y y y y y y	y y y y y y y	y y y y y y y	100
	Testability	y y y y y y y	n y y y y y y	y y y y y y y	95
	Changeability	y y y y y y y	y y y y y y y	y y y y y y y	100
Portability	*Installability*	y y y y y y y	y y y y y y y	y y y y y y y	100
	Deployability	y y y y y y y	y y y y y y y	y y y y y y y	100
Reusability	*Interoperability*	y y y y y y y	y n y y y y y	y y y y y y y	95
	Generality	y y y y y y y	y y y y y y y	y y y y y y y	100
Traceability	*Behavior Trace*	y y y y y -1 y	y y n y y y y	y y y y y -1 y	85
	Controllability trace	y y y y y -1 y	y y n y y y y	y y y y y -1 y	85
Level of Agreement %		100 100 100 100 100 90 100	95 95 85 100 95 100 100	100 100 100 100 100 90 100	

Table 7.5 Level of agreement for software component quality sub-characteristics, Phase I

7.5.1 Case Study

A typical component based web-application (Hong, 2005) is considered to validate and demonstrate the effectiveness of the proposed approach. The illustrative example deals with the component based projects to create ganttcharts. The problem tackled here is that of evaluation and selection of components that can be used for creating ganttcharts. On the lines of classification scheme presented in (Upadhyay and Deshpande, 2010) two components were identified - JGantt (C_x) and GantBiz (C_y). To get the initial ranking of both the components initial quality evaluation considering the 'RUM' aspect of the software component quality model was performed first then it was made available for the evaluation by the evaluators. It is to be noted that the calculation of the *reliability* index is based on the developed quality model in the current chapter, but the concept presented in chapter 6 can also be used to generate *reliability* index. For simplicity measure relative importance between characteristics is considered as equally important i.e. value 0.5. It is to be noted that different value can be considered as per need of project domain. From Table 7.6 to Table 7.10 show the overall computation process for the evaluation of components. The initial evaluation results that C_x is better than C_y, see Table 7.6.

Component	Q_{RUM}	**Ranking**
JGantt	1.250952e+16	I
GantBiz	2.526407e+15	II

Table 7.6 Component ranking based on 'RUM' quality aspect

Characteristics	JGantt	GantBiz
Reliability	13.25	13.25
Usability	3.839544e+08	8.08403e+07
Maintainability	2458924.361	2358628.679

Table 7.7 Component characteristics indices based on 'RUM' quality aspect.

Characteristics – Reliability		JGantt	GantBiz
Service Maturity	**Recoverability**	1	1
	Error prone	2	2
	Error handling	2	2
Outcome Set	**Correctness**	4	4
Reliability Index (I_R)		13.25	13.25

Table 7.8 Component reliability index (based on SCQM)

Characteristics – Maintainability		JGantt	GantBiz
	Parameterization	4	4
	Adaptability	3	3
	Change control capacity	3	3
	Priority	2	2
Customizability		88.812	68.562
	Start up self test	5	5
	Trail version	5	5
	Test suite provided	4	4
	Test material	3	3
Testability		331.562	331.562
Changeability	**Upgradeablity**	4	5
	Debugging	4	4
	Backward compatibility	5	5
		83.5	104.75
Maintainability Index (I_M)		2458924.361	2358628.679

Table 7.9 Component maintainability index (based on SCQM)

Characteristics – Usability		JGantt	GantBiz
	Help system	5	4
	Manuals	4	4
	Tutorials and Demos	4	4
	Support tools and services	1	1
Help mechanisms		101.31	82.812
	Time to use	5	4
	Time to configure	5	3
	Time to administer	5	4
Learnability		129	51
	Operation effort	4	4
	Administration effort	4	4
	Customizability effort	5	4
Operability		83.5	67.25
	Directory listing	4	4
	Search and fetch	4	4
	Classification	4	4
	Marketing information	5	4
Approachability		351.812	284.562
Usability Index (I_U)		3.839544e+08	8.08403e+07

Table 7.10 Component Usability index (based on SCQM)

For the quality evaluation of the components as the requirement of phase II, the groups were asked to apply the quality evaluation technique (section 7.4) for selecting a candidate component based upon '*RUM*' quality characteristics. The result of the quality evaluation of all the groups indicated that the C_x component were the most appropriate for selecting and integrating to the system. This result matched with the initial result. The groups were then asked to give their opinion on the developed software component

quality model and assessment technique. They were also questioned if they would adopt the same. Both the negative and positive answers were recorded. Table 7.11 shows their answers and comments.

More specifically Group 'R' and Group 'D' were extremely satisfied with the developed software component quality model and quality evaluation technique. Some people of Group 'A' found little difficulty in carrying out the evaluation. This happened because of lack of prior experience on their part to evaluate comprehensively quality of components.

Group		Time to learn and use the methodology (man hours)	Time/effort spent man (hours)	Comment	Opinion	Best Alternative
Group 'R'	R_1	1.5	6	Good approach and accurate	Will adopt	C_x
	R_2	2.3	6.5	Systematic approach and accurate	Will adopt	C_x
	R_3	2.2	5.5	Good approach and accurate	Will adopt	C_x
	R_4	1.5	5.5	Good approach and accurate	Will adopt	C_x
	R_5	2.2	6.2	Systematic approach, better than existing approaches	Will adopt	C_x
	R_6	3	6.5	Systematic approach, better than existing approaches	Will adopt	C_x
	R_7	2.5	6.2	Systematic approach, better than existing approaches	Will adopt	C_x
Group 'A'	A_1	2.4	7.5	Difficult at first but useful	Will adopt	C_x
	A_2	3	9.5	Better than current practice, but time consuming and need more effort	Will not adopt	C_x
	A_3	3	7	Difficult at first but useful	Will adopt	C_x
	A_4	2.5	7.5	Difficult at first but useful	Will adopt	C_x
	A_5	3.5	8	Better than current practice, but need more effort	Will adopt	C_x
	A_6	4	8	Better than current practice, but need more effort	Will adopt	C_x
	A_7	3.5	8.2	Better than current practice, but time consuming and need more effort	Will not adopt	C_x
Group 'D'	D_1	1.4	5.5	Good approach and accurate	Will adopt	C_x
	D_2	2	6.5	Systematic approach	Will adopt	C_x
	D_3	2	5	Good approach and accurate	Will adopt	C_x
	D_4	1.5	5.5	Good approach and accurate	Will adopt	C_x
	D_5	1.5	5.2	Systematic approach	Will adopt	C_x
	D_6	3	5.3	Systematic approach	Will adopt	C_x
	D_7	2.5	5.2	Systematic approach	Will adopt	C_x

Table 7.11 Overall validation results

Finally, satisfaction and confidence level of all the groups shows the desired effectiveness of developed model and methodology, see from Figure 7.4 to Figure 7.6. In the global market, the adoption of component quality model and methodological framework to evaluate components quality provides the individual (user, group or organization) an edge over the competitors.

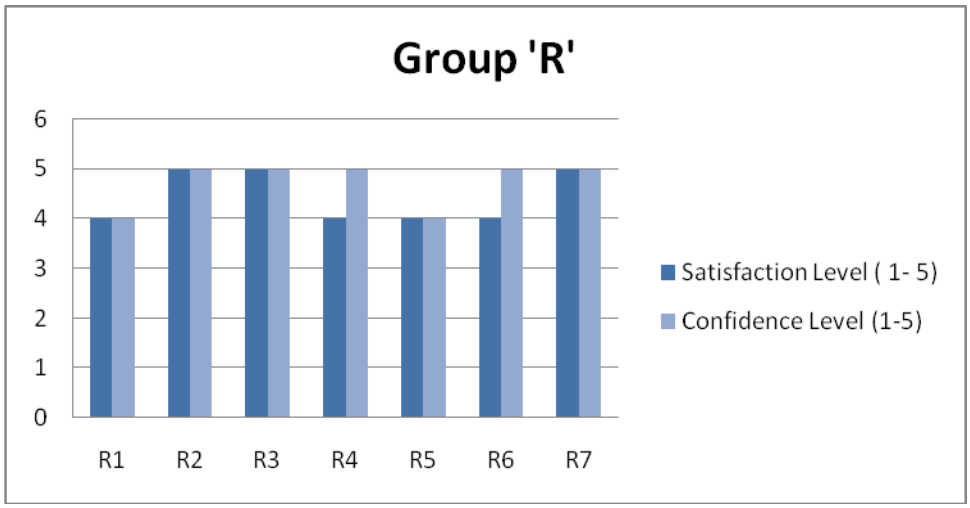

Figure 7.4 Group ‘R’ satisfaction level and confidence level for quality model and evaluation tool.

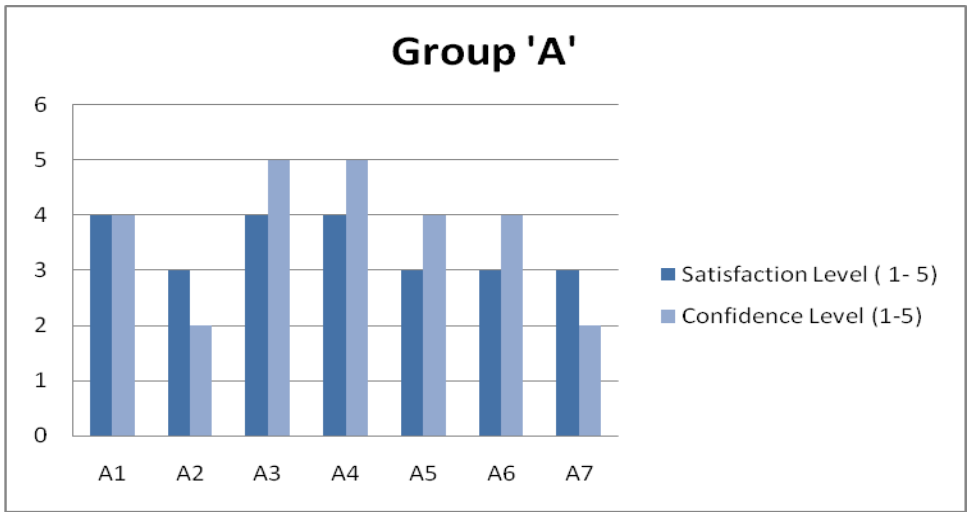

Figure 7.5 Group ‘A’ satisfaction level and confidence level for quality model and evaluation tool.

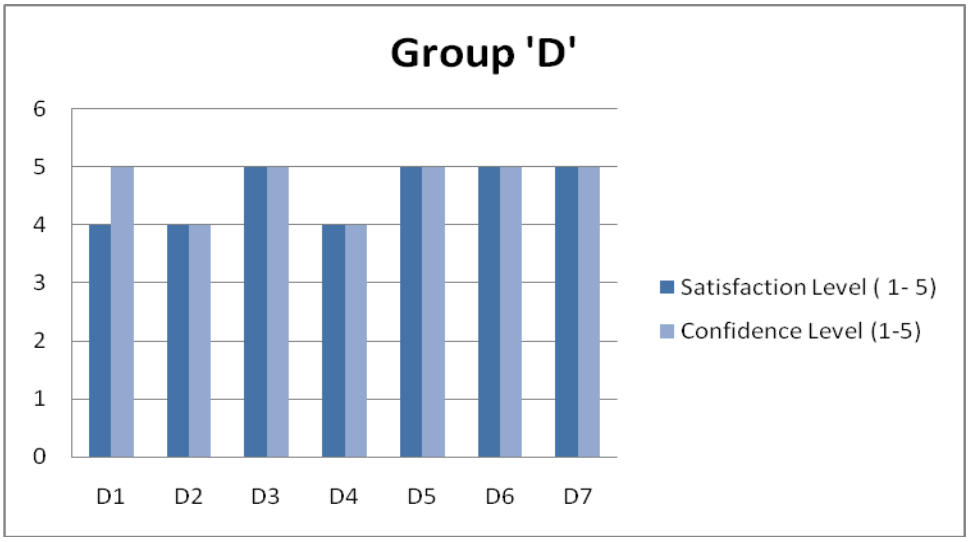

Figure 7.6 Group 'D' satisfaction level and confidence level for quality model and evaluation tool.

7.6 Concluding Remarks

In this chapter, the software component specific quality model is developed. The quality model is a first step towards research effort in the direction of building up a software component quality assurance framework. The model is designed in such a way as to overcome the drawbacks of the existing models. A graph theoretic methodological framework is also developed to evaluate the quality of components considering all characteristics concurrently. The developed methodology and framework is validated with case study. Component quality is represented as a single expression using permanent matrix (one-to-one mapping to digraph) and permanent function. Quality index is also developed that can be used for the selection and the ranking of candidates.

In the next chapter, the software component selection techniques in fuzzy as well as non-fuzzy environments have been developed and discussed. The case study is also presented to show the applicability of the developed techniques.