QUALITY ATTRIBUTES

Software quality plays an important role in success of the overall software system. It is considered as a very important aspect if the system to be built for all stakeholders. Software quality is the extent to which product defined set of desirable features are incorporated into a product so as to enhance its marketing [78]. For any software system there must be following three specifications such as functional specification (what system is to do), quality specification (how well the functions are to operate), resource specification (how much is to be spent on the system). Quality comprises all characteristics and significant features of a product or an activity which relate to the satisfying of given requirements [79]. This chapter starts with the definitions of software quality and quality attributes. Various quality models widely used in industries are discussed in this chapter. A discussion on the quality attributes used in this research is presented.

3.1 DEFINITIONS OF SOFTWARE QUALITY AND QUALITY ATTRIBUTE

There are number of ways researchers define quality and its attributes. Following are a few example definitions from research literature for quality.

According to Industry gurus:

- Crosby defines quality as a conformance to specifications [80].
- According to Deming, quality must be defined in terms of customer satisfaction Deming insists that quality should be defined only in terms of the judge of quality. He stresses that everyone in the industry should try for meeting and exceeding the customers' requirements [81].
Ishikawa’s perspective on quality is a “meeting customer needs” definition as he relates the level of quality to every changing customer expectations. He quotes that as the requirements and the expectations of a customer change continuously, quality is a dynamic concept. So, quality must be defined comprehensively and dynamically. Ishikawa also comments the impact of cost on customer satisfaction. Overprized product will not get customer satisfaction [82].

According to Juran, the word quality has multiple meanings. Two of those meanings are as follows 1) Quality consists of those product features which meet the need of customers and thereby provide product satisfaction. 2) Quality consists of freedom from deficiencies. He defines shortly quality as fitness for use [83].

**According to standard bodies**

- ANSI/ASQC A3/1978 defines Quality as the total of features and characteristics of a product or a service that bears on its ability to satisfy the given needs.
- According to IEEE Software Quality is
  1. The degree to which a system, component, or process meets specified requirements.
  2. The degree to which a system, component, or process meets customer or user needs or expectations [84].

**Software Quality Attributes** are the benchmarks that describe system’s intended behavior within the environment for which it was built. The quality attributes provide the means for measuring the fitness and suitability of a product. Software architecture has a profound affect on most qualities in one way or another, and software quality attributes affect architecture.
Much of a software architect’s life is spent designing software systems to meet a set of quality attribute requirements [85]. General software quality attributes include scalability, security, performance and reliability. Quality attribute requirements are part of an application’s nonfunctional requirements, which capture the many facets of how the functional requirements of an application are achieved. All but the most trivial application will have nonfunctional requirements that can be expressed in terms of quality attribute requirements.

Understanding trade-offs between quality attribute requirements, and designing a solution that makes sensible compromises is one of the toughest parts of the architect role. It’s simply not possible to fully satisfy all competing requirements. It’s the architect’s job to tease out these tensions, make them explicit to the system’s stakeholders, prioritize as necessary, and explicitly document the design decisions.

Software Quality Attribute Trade-offs Designers need to analyze trade-offs between multiple conflicting attributes to satisfy user requirements. The ultimate goal is the ability to quantitatively evaluate and trade off multiple quality attributes to arrive at a better overall system. Number of quality attributes has been proposed based on clustering of quality attributes. These models are discussed in detail in the following section.

3.2 SOFTWARE QUALITY MODELS

Quality models proposed in the literature are as follows:

- McCall
- Boehm
- ISO 9216
- Dromey
McCall’s Quality Model

The widely known and the first quality model is McCall model that was originated from the US military. The McCall quality model has the following three perspectives to define and identify the quality of a software product as shown in fig 3.1. They are product revision (ability to undergo changes), product transition (adaptability to new environments) and product operations (its operation characteristics) [86].

![McCall's Quality Model Diagram]

**Fig 3.1 McCall’s Quality model**

This quality model has a number of quality factors as shown in fig 3.2 that reflect the user’s views and developer’s priorities to bridge the gap between them.
Boehm’s model

Boehm proposed a quality model that is the second of the basic quality models. Boehm deals the issues of models that automatically and quantitatively evaluate the quality of software. In essence his models attempts to qualitatively define software quality by a given set of attributes and metrics [87]. Like McCall’s model, Boehm's model also presents a hierarchical quality model of high-level characteristics, intermediate level characteristics and primitive characteristics as depicted in fig 3.3.
Fig 3.3 Boehm’s model

ISO 9126 model

This model is published as ISO 9126: Software Product Evaluation: Quality Characteristics and Guidelines for their Use-standards. McCall and Boehm models are the basis for this standard. This model is organized as like McCall and Boehm models. Similarly, ISO 9126 also includes functionality as a parameter, as well as identifying both internal and external quality characteristics of software products as shown in fig 3.4.
Dromey proposes a product based quality model. It understands clearly that each system requires different type of quality evaluation. Hence, it needs a more dynamic idea for modeling the process that is applicable for different systems. Dromey focuses mainly on the relationship between the quality attributes and the sub-attributes and between the software product properties and software quality attributes as shown in fig 3.5. Product quality depends on the tangible properties of components and component composition [88].
3.3 QUALITY ATTRIBUTES

List of quality attributes considered in the research for balancing the quality attributes during software architecture evaluation process are described below. Quality attributes like maintainability, testability, portability, flexibility, reusability, simplicity, availability, security, performance, concurrency, reliability, scalability, cost, life time, usability and usability are considered in this research, as these attributes are widely used in the research literature for comparative analysis of the results [89] [90]. These are also the attributes used by architect during architecting process [90].

1. Maintainability:

Maintainability is the ability of the system to undergo changes with a degree of ease. These changes could impact components, services, features, and interfaces when adding or changing the functionality, fixing errors, and meeting new business requirements. Software maintainability, the ease with which a software system can be modified [89], is an important software quality attribute. Maintenance process is associated with this quality attribute. This process consumes majority of the costs of a Software Development Life Cycle [91]. Henceforth, software costs have a great impact by the maintainability of a software system. If this cost is predicted earlier, it is helpful to manage the whole project cost.

Maintainability is the extent to which software is capable of being changed after deployment. Software may need to be modified to fix remaining errors, deal performance issues and to deal the changes in software requirements.

2. Testability

Testability is a measure of how easy it is to create test criteria for the system and its components, and to execute these tests in order to determine if
the criteria are met. Good testability makes it more likely that faults in a system can be isolated in a timely and effective manner.

**Software testability** is the degree to which a software artifact (i.e. a software system, software module, requirements- or design document) supports testing in a given test context. If the testability of the software artifact is high, then easily faults can be found in the system [92]. If the testability of the software artifact is low, then the test effort to be taken is high.

3. **Concurrency**

Concurrency refers the degree of supporting simultaneous interactions or computations without degrading the performance of the system. The challenges of designing concurrent systems arise mostly because of the interactions which happen between concurrent activities. When concurrent activities interact, some sort of coordination is required.

4. **Simplicity**

Simplicity is a desirable quality attribute in any software system. Software system should be simple to use without overcrowded, non usable widgets. Designing a simple system is time consuming. It will have little impact with the architecture level.

5. **Security**

Security is the ability of a system to prevent malicious or accidental actions outside of the designed usage, and prevent disclosure or loss of information [93]. Security ensures the reliability of the system. Securing a system should protect assets and prevent unauthorized access to or modification of information. The features used to secure systems are authentication, encryption, auditing, and logging.
6. **Supportability**

Supportability is the ability of the system to provide useful information to recognize and solve issues when it fails to work correctly. Supportability should involve with diagnosis, troubleshoot and traceability [89].

7. **Usability**

Usability is the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use [94]. Usability refers how easy and pleasant to use the features of the system [95], whereas utility refers whether the system provides the features that user required.

8. **Portability**

Portability refers to the ability of the system or part of the system to move across environments. If the product can be made to run as a whole on different environments, it will reduce cost and time. Every four years, a new hardware is purchased by the client industry. So the client industry should be able to run the existing software on a new hardware with less expensive. A portable product reduces the cost to run on the new computer than to write a new product from scratch.

9. **Flexibility**

Flexibility refers the capacity of a system to be adapted for different environments and situations in order to face changes related to business environment policies and rules. Flexibility improves the quality of the system within its development life cycle. It promises the stability between the artifacts of following phases of development life cycle.

10. **Reusability**

A system is portable one if it runs in a different environment after few modifications. Whereas Reusability of a system refers the usage of its
components develop a different system with different functionality. When a new system is built by using the components of previously developed system, then it is referred as opportunistic reusability. If the system is built using components that are constructed specifically for future reuse, then it is termed as systematic reusability.

Repeated use and test will produce higher quality products. Every time the component is reused in developing a new system, it will be tested while testing a system. Hence the reliability level and usefulness of the component increases. It also reduces the development time, and therefore cost.

11. Performance

Performance is the degree to which a system or component accomplishes its designated functions within given constraints, such as speed, accuracy, or memory usage [89]. It also refers to responsiveness [96], either the time required to respond to specific events or the number of events processed in a given interval of time. Performance is that attribute of a computer system that characterizes the timeliness of the service delivered by the system.

12. Availability

The availability of a system is a measure of its readiness for usage. Availability is always a concern when considering a system’s dependability, though to varying degrees, depending upon the application.

Availability is measured as the limit of the probability that the system is functioning correctly at time t, as t approaches infinity. This is the steady-state availability of the system. It may be calculated as $MTTF/(MTTF + MTTR)$ where MTTF is the mean time to failure, and MTTR is the mean time to repair.
13. **Reliability**

Reliability is the ability of a system to remain operational over time. Reliability is measured as the probability that a system will not fail to perform its intended functions over a specified time interval. The reliability of a system is typically measured as its mean time to failure (MTTF), the expected life of the system [97].

14. **Cost**

Cost involves with the expense of building, maintaining and operating the system.

15. **Lifetime**

The Period of the time that the product is alive before retirement.

16. **Scalability**

Scalability is ability of a system to handle the extra load without degrading the performance of the system. Scalability can be achieved either by increasing the number of components or number of systems [89]. The key issues for scalability are:

- Inability to handle increasing load by the application
- Response and completion time may be extended
- Inability to process and queue the additional works.

Quality plays a dominant role during the architecting process. Quality attributes are collected during the requirements elicitation process and documented in SRS. During architecture evaluation process, architect with his team needs to evaluate quality attributes with reference to SRS. The next chapter deals with the existing software architecture analysis techniques, with advantages and disadvantages.