# UNIT-I

# SOFTWARE QUALITY

**What is Software Quality?**

It is difficult to define software quality, as the answer depends on several aspects like kind of software, who is defining and his role, and circumstances bit it is easy to define the characteristics of high-quality software. To understand the characteristics of high-quality software, we need to understand different perspectives and expectations of users as well as other people involved with the development, management, marketing, and maintenance of the software products.

**Quality: Perspectives and expectations**

1. **Transcendental view:** In this view, quality is hard to define or describe in abstract terms but can be recognized if it is present. It is associated with some intangible properties that delight users.
2. **User view:** In this view, quality means conformance to process standards.
3. Manufacturing view: In this view, quality means conformance to process standards.
4. **Product view:** In this view, the focus is on inherent characteristics in the product itself in the hope that controlling these internal quality indicators will result in improve external product behaviour.
5. **Value-based view:** In this view, quality is the customers’ willingness to pay for any software.

**People’s roles and responsibilities**

We consider two broad categories of people:

1. Customers
2. Producers

**Customers:** customers can be the people who wanted the software to be developed like Banks, Application owners who wants the software for their businesses or the users who will directly use the software in their regular activities like bank employees, online customers, or product users.

**Producers:** producers are the people who are involved in the software development like software developers, testers, QA team, marketing people, maintenance, services, add-on services, packaging, certification, and independent verification and validation (IV & V) etc.

Even though there are some common views in both the groups, we use external view (Black box) for the customers group’s perspective as they can only observe the external behaviour of the software but can’t see the internal factors which lead to that behaviour. And we use internal view (White box) for the producers group’s perspective as they can observe the internal characteristics of the software.

**Quality expectations on the Customer side:**

The basic quality expectations of a customer is that the software must perform all the functionalities which are specified and all those functionalities must be reliable means work as expected over long-term/repeated use. For today’s users ease of use and usability is more important quality expectation than reliability. For example, people choose GUI over text editors. In the same way, ease of installation is another concern; people choose plug-n-play kind of software over lengthy installation procedures. Different users of the same system may have different views and priorities, like for a web application, sophisticated user concerned about reliability and a non-sophisticated user concerned about usability. Sometimes the user can be a non-human user who needs an interactive system with better inter-operability and adaptability. In addition with the basic quality expectations, customer needs a budget friendly software or service which fulfils the value-based view of the quality.

**Quality expectations on the producer side:**

The basic quality expectation for a producer is to meet the contractual specifications and service agreement terms. Various product internal characteristics that make it easy to conform to product specifications, such as good designs that maintain conceptual integrity of product components and reduce coupling across different components, are also associated with good quality. For product and service managers, adherence to pre-selected software process and relevant standards, proper choice of software methodologies, languages, and tools, as well as other factors, may be closely related to quality. They are also interested in managing and satisfying user’s quality expectations, by translating such quality expectations into realistic quality goals that can be defined and managed internally, selecting appropriate and effective QA strategies, and seeing them through. For other people on the producer side, their different concerns may also produce quality views and expectations different from the above. For example, usability and modifiability may be paramount for people involved with software service, maintainability for maintenance personnel, and portability for third-party or software packaging service providers, and profitability and customer value for product marketing.

**QUALITY FRAMEWORKS AND ISO-9126:**

ISO-9126 (ISO, 2001) is the mostly influential quality framework in the software engineering community today. ISO-9126 provides a hierarchical framework for quality definition, organized into quality characteristics and sub-characteristics.

There are six top-level quality characteristics, with each associated with its own exclusive (non-overlapping) sub-characteristics, as summarized below:

1. **Functionality:** A set of attributes that depend on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs. The sub-characteristics are:
   1. Suitability
   2. Accuracy
   3. Interoperability
   4. security
2. **Reliability:** A set of attributes that depend on the capability of software to maintain its level of performance under specified conditions for a specified period of time. The sub-characteristics are:
   1. Maturity
   2. Fault tolerance
   3. recoverability
3. **Usability:** A set of attributes that depend on the effort needed for use, and on the individual assessment of such use, by a specified or implied set of users. The sub-characteristics are:
   1. Understandability
   2. Learnability
   3. Operability
4. **Efficiency:** A set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used, understand conditions. The sub-characteristics are:
   1. Time behaviour
   2. Resource behaviour
5. **Maintainability:** A set of attributes that depend on the effort needed to make specified modifications. The sub-characteristics include:
   1. Analysability
   2. Changeability
   3. Stability
   4. Testability
6. **Portability:** the set of attributes that depend on the ability of software to be transferred from one environment to another. The sub-characteristics are:
   1. Adaptability
   2. Installability
   3. Conformance
   4. Replaceability

**Alternate Frameworks and Focus on Correctness:**

ISO-9216 framework describes attributes and properties which associate with quality. The hierarchy has sub-characteristics which are not sharing any quality characteristics with each other. But sometimes the product properties may be linked with multiple quality characteristics or sub-characteristics. For example, various forms of redundancy may affect both efficiency and maintainability. So, various alternative frameworks are proposed to have more flexible relations among different quality attributes, and for a smooth transition from specific quality concerns to specific product properties or metrics.

Many companies and communities adapted and customized existing quality frameworks to define quality for themselves.

1. For example, the quality attribute list CUPRIMDS (capability, usability, performance, reliability, installation, maintenance, documentation, and service) IBM used for their software products. CUPRIMDS is often used together with overall customer satisfaction (CUPRIMDSO) to characterize and measure software quality for IBM’s software products.
2. A set of quality attributes has been identified for web-based applications, with the primary quality attributes as reliability, usability, and security, and the secondary quality attributes as availability, scalability, maintainability, and time to market.
3. For example, performance (or efficiency) and reliability would take precedence over usability and maintainability for real-time software products. On the contrary, it might be the other way round for mass market products for end users.

**Correctness:**

Correctness is typically related to several quality characteristics or sub-characteristics in quality frameworks. Correctness is still a fundamental part of the users’ expectations. Key to the correctness aspect of software quality is the concept of defect, failure, fault, and error. The term “defect” generally refers to some problem with the software, either with its external behaviour or with its internal characteristics.

* **Failure:** The inability of a system or component to perform its required functions within specified performance requirements. It refers to a behavioural deviation from the user requirement or the products specification.
* **Fault:** An incorrect step, process, or data definition in a computer program. It refers to an underlying condition within software that causes certain failure to occur.
* **Error:** A human action that produces an incorrect result. It refers to a missing or incorrect human action resulting in certain fault(s) being injected .into software. Errors include error sources, or the root causes for the missing or incorrect actions, such as human misconceptions, misunderstandings, etc.

**Defects:** Failures, faults, and errors are collectively referred to as defects.

|  |
| --- |
|  |

The concepts of error, fault, failure, and defect can be placed into the context of software artifact, software development activities, and operational usage as shown in the above diagram.

1. **Left box:** The inputs to the software development activities (conceptual models and information, developers with certain knowledge and experience, reusable software components, etc.) are having some problems which are given for development.
2. **Middle box:** The software system as represented by its artifacts is depicted in the middle box. The artifacts include mainly software code (and designs, specifications, requirement documents, etc.) are having faults.
3. **Right box:** Usage scenarios and execution results are either failures or deviated from the expected results.
4. The errors as missing or incorrect human actions are not directly depicted within one box, but rather as actions leading to the injection of faults in the middle box because of some error sources in the left box.

A causal relation exists among these three aspects of defects:

errors -> faults -> failures

That is, errors may cause faults to be injected into the software, and faults may cause failures when the software is executed. However, this relationship is not necessarily 1-to- 1: A single error may cause many faults, such as in the case that a wrong algorithm is applied in multiple modules and causes multiple faults, and a single fault may cause many failures in repeated executions. Conversely, the same failure may be caused by several faults, such as an interface or interaction failure involving multiple modules, and the same fault may be there due to different errors.

**The Correctness-centered properties and measurements:**

The correctness-centered quality from the external view, or from the view of consumers (users and customers) of a software product or service, can be defined and measured by various failure-related properties and measurement. To a user or a customer, the primary concern is that the software operates without failure, or with as few failures as possible. When such failures or undesirable events do occur, the impact should be as little as possible. These concerns can be captured by various properties and related measurements, as follows:

* **Failure properties and direct failure measurement:** Failure properties include information about the specific failures, what they are, how they occur, etc. These properties can be measured directly by examining failure count, distribution, density, etc.
* **Failure likelihood and reliability measurement:** How often or how likely a failure is going to occur is of critical concern to software users and customers. This likelihood is captured in various reliability measures, where reliability can be defined as the probability of failure-free operations for a specific time period or for a given set of input.
* **Failure severity measurement and safety assurance:** The failure impact is also a critical concern for users and customers of many software products and services, especially if the damage caused by failures could be substantial. Accidents, which are defined to be failures with severe consequences, need to be avoided, contained, or dealt with to ensure the safety for the personnel involved and to minimize other damages.

**Evolving perceptions of quality:**

Before software and information technology (IT) industries came into existence, quality has long been associated with physical objects or systems, such as cars, tools, radio and television receivers, etc. The QA focus is on ensuring that the products conform to their specifications. These specifications often accompany the finished products, so that the buyers or users can check them for reference. For example, the user’s guide for stereo equipment often lists their specifications in terms of physical dimensions, frequency responses, total harmonic distortion, and other relevant information. In addition to these specifications, some products even mention error tolerance, reducing variance kind of statistical quality control specifications. For example, the initial quality for automobiles is defined to be the average number of reported problems per 100 vehicles by owners during the first three years. Another commonly used quality measure for automobiles, reliability, is measured by the number of problems over a longer time for different stages of an automobile’s lifetime. With the development of service industries, an emerging view of quality is that business needs to adjust to the dynamically shifting expectations of customers, with the focus of quality control shifting from zero defects in products to zero defection of customers. According to software industry has incorporated both the conformance and service views of quality, and high-quality software can be defined by three basic elements: conformance, adaptability, and innovation.

**Quality in software engineering:**

The primary concerns for quality, were used to divide software engineering into four progressive stages:

1. In the **functional stage**, the focus was on providing the automated functions to replace what had been done manually before.

2. In the **schedule stage**, the focus was on introducing important features and new systems on a timely and orderly basis to satisfy urgent user needs.

3. In the **cost stage**, the focus was on reducing the price to stay competitive accompanied by the widespread use of personal computers.

4. In the **reliability stage**, the focus was managing users’ quality expectations under the increased dependency on software and high cost or severe damages associated with software failures.

We can see a gradual increase in importance of quality within software engineering. And the importance of focusing on correctness-centered quality attributes in our software QA effort for modern software systems.