SOFTWARE QUALITY

→ Inorder to define software quality, we need to examine the different perspectives and expectations of users as well as other people involved with the development, management, marketing and maintenance of the software products.

→ Five major views according to (Kitchenham and Pfleeger, 1996; Pfleeger et al., 2002) are:

1) Transcendental view -
In this view, quality is hard to define is abstract terms but can be recognized if it is present.

2) User view -
Quality is fitness for purpose or meeting user's needs.

3) Manufacturing view -
Quality means conformance to process standards.

4) Product view -
The focus is on inherent characteristics in the product itself in the hope that controlling these internal quality indicators will result in improved external product behavior.

5) Value-based view -
Quality is customers willingness to pay for a software.

→ Different people will have different views of quality based on their roles and responsibilities.

→ people can be divided into 2 categories with focusing on QA and quality engineering.

1) Consumers
It includes customers who are responsible for the acquisition of software products or services and users who use the software product.

2) Producers
It includes anyone involved in development, management, maintenance, marketing and service of software product. Third party participants involved in add-on products & services, software packaging, software certification etc. are also producers.

Quality expectations on the consumer side:-

→ The basic quality expectations of a user are that a software system performs useful functions as it is specified. There are 2 basic elements to this expectation:
(i) it performs right functions as specified that fits the user's needs
(ii) it performs these specified functions correctly over repeated use or over a long period of time, or performs its functions reliably.

→ For many users of today's ubiquitous software and systems, ease of use or usability may be more important quality expectation then reliability.

→ Ease of installation is another major trend for software intended for today's population.

Quality expectations on the producer side:-

→ For software producers, the most important quality question is to fulfil their contractual obligations by producing software products that conform to product specifications or providing services that conform to service agreement.

→ 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.

→ usability and modifiability may be paramount for people involved with software service, maintainability for maintenance personnel, portability for third-party or software packaging service providers and profitability and customer value for product marketing.

Quality framework and ISO 9126:-

ISO 9126 provides a hierarchical framework for quality definition. There are 6 top-level quality characteristics, with each associated with its own exclusive sub-characteristics.

1] Functionality - A set of attributes that bear on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs.

The subcharacteristics include:

a) suitability - refers to the appropriateness of the functions of the software.

b) Accuracy - refers to the correctness of the functions.

c) Interoperability - This subcharacteristic concerns the ability of a software component to interact with other components or systems.

d) Security - relates to unauthorized access to the software functions.

2] Reliability - A set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time. The sub-characteristics include:

(a) Maturity - concerns frequency of failure of the software.
(b) Fault tolerance - The ability of software to withstand and recover from component, or environmental failure.
(c) Recoverability - ability to bring back a failed system

3] Usability :- A set of attributes that bear on the effort needed for use and on the individual assessment of such use, by a stated or implied set of users.
The sub-characteristics include:
a) Understandability - Determines the ease of which the system functions can be understood, relates to user mental models in human computer Interaction methods.
b) Learnability - Learning effort for different users, i.e novice, expert, casual etc.
c) Operability - ability of the software to be easily operated by a given user in a given environment.

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, under stated conditions.
The sub-characteristics include:
a) Time behavior - Characterizes response times for a given throughput, i.e. transaction rate.
b) Resource behavior - characterizes resources used, i.e memory cpu, disk and network usage.

5] Maintainability - A set of attributes that bear on the effort needed to make specified modifications.
The sub-characteristics include:
a) Analyzability - characterizes the ability to identify the root cause of a failure within the software.
b) Changeability - characterizes the amount of effort to change

a system.

c) Stability - characterizes the sensitivity to change of a given system that is the negative impact that may be caused by system changes.

d) Testability - Characterizes the effort needed to verify (test) a system change.

6] Portability - A set of attributes that bear on the ability of software to be transfered from one environment to another. The sub-characteristics include:

a) Adaptability - characterizes the ability of the system to change to new specifications or operating environments.

b) Installability - Characterizes the effort required to install the software.

c) Conformance - similar to compliance for functionality, but this characteristic relates to portability.

d) Replaceability - characterizes the plug and play aspect of software components, that is how easy is it to exchange a given software component within a specified environment.

* Correctness and Defects : Definitions, properties and Measurements

→ Failure: The inability of a system or component to perform its required functions within specified performance requirements. It refers to a behavioral deviation from the user requirement or the product specification.

→ Fault: An incorrect step, process or data definition in a computer program. It refers to an underlying condition within a software that causes certain failures to occur.

→ Error: A human action that produces an incorrect result. It refers to a missing or incorrect human action resulting

in certain faults being injected into a software.

→ defects: Failures, faults and errors are collectively referred to as defects (also refered to as "bugs").

Properties and measurements:-

1) Failure properties and direct failure measurement:
- Failure properties include information about the specific failures like, what they are, how they occur etc.
- We can measure these properties by examining failure count, distribution, density etc.

2) Failure likelihood and reliability measurement:
- Failure likelihood refers to how likely or how often a failure is going to occur. 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.

3) Failure severity measurement and safety assurance:
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.

* fault properties and related measurements:
- Individual faults can be analyzed and examined according to their types, their relations to specific failure and accidents, their causes, the time and circumstances when they are injected etc.
- Faults can be analyzed collectively according to their distribution and density over development phases and different software components.

Defects in context of QA:

- Ensuring quality means dealing with defects
- 3 generic way of dealing with defects
  (i) defect prevention
  (ii) defect detection and removal
  (iii) defect containment.

- In the process of dealing with defects, various direct defect measurements and other indirect quality measurements can be taken which will form a multi-dimensional measurement space referred to as quality profile. Using various models on these measurement results need to be analyzed to provide quality assessment and feedback to the overall software development process.

Quality engineering

- can be viewed as defect management
- includes:
  (i) quality planning before specific QA activities are carried out
  (ii) execution of planned QA activities.
  (iii) measurement, analysis and feedback to monitor & control the QA activities.