Unit-1

-(1.1)

Define Quality.

Quality means conformance to requirements. It consists of those product features which meet the needs of customers and thereby provide product satisfactions.

What is Software Quality?

Software Quality is defined as a degree to which a system, component or process conforms to explicitly states both **functional and non-functional requirements**, documented development standards and other characteristics expected by a properly developed software.

Software quality is defined as a field of study and practice that describes the desirable attributes of software products. There are two main approaches to software quality: defect management and quality attributes.

-(1.2)
-Product Operation Software Quality Factors

Software Quality Factors and Models

All the softwares may satisfactorily fulfill the basic requirements for correct calculations but suffered from poor performance in important areas such as maintenance, reliability, software reuse, or training. The cause for the poor performance of the developed software projects in these areas was the lack of predefined requirements to cover these important aspects of the software's functionality. There is a need for a

comprehensive definition of requirements that will cover all the attributes of software and aspects of the use of software, including usability aspects, maintainability aspects, and so forth in order to assure the full satisfaction of the users.

The great variety of issues related to the various attributes of software and its use and maintenance, as defined in software requirements documents, can be classified into content groups called **quality factors.** It can also be defined as various factors which influence the software. Several models of software quality factors and their categorization in factor categories have been suggested over the years.

McCall's factor model

McCall's factor model classifies all software requirements into 11 software quality factors. The 11 factors are further grouped into three categories —

Product operation factors (deals with the daily operations of the software)

- **Correctness**
  The extent to which a software **meets its requirement specification**.

- **Reliability**
  The extent to which a software **performs its intended task without failure.**

- **Efficiency**

  The hardware resources needed to perform all the functions of the software system in conformance to all other requirements.

- **Integrity**

  The extent to which the software can **prevent unauthorized users from accessing the data or the system.**

- **Usability**

  It deals with the **scope of efforts required to learn, operate and understand the functions of a software.**

Product revision factors (deals with requirements of testing and maintenance)

- **Maintainability**

  The effort required to detect and correct an error during the maintenance phase.

- **Flexibility**

  This factor deals with the capabilities and efforts required to support adaptive maintenance activities of the software

- **Testability**

  The efforts needed to verify a software to ensure that it meets the specified requirements.

Product transition factors (deals with the adaption to other environments)

- **Portability**

  The efforts needed to transfer a software from one platform to another consisting of different hardware, OS etc.

- **Reusability**

  The extent to which the **programmer's code can be reused in other applications being developed.**

- **Interoperability**

  The extent to which an interface can be developed that **enables communication with other software.**

Other SQF models

The Evans and Marciniak Factor Model

The factors are the same as McCall's 11 factors but excluding testability factors. 12 quality factors divided into three sub groups: **Design, Performance and Adaptation.** Two more quality factors are added:

Verifiability Design)

Defining design and programming features in such a way that enables its efficient verification. It refers to modularity, simplicity, and adherence to documentation and programming guidelines.

Expandability Adaptation)

It refers to the future efforts that will be needed to serve large populations, improve service, or add new applications in order to improve usability.

- **Design**

  Correctness

  Maintainability

  Verifiability

- **Performance**

  Usability

  Reliability

  Integrity

  Efficiency

- **Adaptation**

  Portability

  Flexibility

  Reusability

  Interoperability

  Expandability

The Deutsch and Willis Factor Model

The factors are the same as The Evans and Marciniak Factor model but add three more quality factors. 15 quality factors divided into four subgroups: **Functional, Performance, Change, Management**

1)Safety Performance)

The extent to which hazardous conditions can be eliminated for the user as a result of errors in process control software.

2)Manageability Management)

It refers to the administrative tools that support software modification during the development and maintenance phase.

3)Survivability Functional)

The extent to which the service provided by the software can be continued. This also defines the minimum time allowed between failures of the system, and the maximum time permitted for recovery of service.

- **Functional**

  Survivability

  Reliability

  Usability

  Integrity

- **Performance**

  Efficiency

  Safety

  Correctness

  Interoperability

- **Change**

  Reusability

  Maintainability

  Portability

  Flexibility

- **Management**

  Manageability

  Expandability

  Verifiability

-(1.3)
-Quality Assurance Vs Quality Control: What's the Difference?

Unit-2

-(2.1)
-
https://www.google.com/amp/s/www.geeksforgeeks.org/software-engineering-software-quality-assurance/amp/

-(2.2)

Software Reviews & Audits

A review is a systematic examination of a document by one or more people with the main aim of finding and removing errors early in the SDLC. The review is generally done by a committee composed of the project leader, project manager, client representative and a software engineer. Reviews are used to verify documents such as requirements, systems design, code, test plans and test cases. Reviews are usually performed manually.

Importance of reviews:

- Productivity of the development team is improved and timescales reduced because of the correction of defects in early stages and work products are clear and unambiguous.

- Testing costs and time is reduced as there is enough time spent during the initial phase.

- Reduction in costs because fewer defects in the final software.

- It can be used to train technical authors in the development of extremely low-defect documents, and also to identify and remove process inadequacies that encourage defects

There are mainly three types of software reviews:

- **Software Peer Review**

  Peer review is a process of assessing the technical content and the quality of the product usually done by the author of the work along with the colleague developers. It is performed in order to examine or resolve the defects in the software, whose quality is also checked by other members of the team.

- **Software Management Review**

  These types of reviews are carried out by representatives at a managerial level to evaluate the status of the work done and to make decisions regarding the downstream activities. They are carried out by, or on behalf of, the management personnel having direct responsibility for the system. These reviews identify consistency with and deviations from plans

- **Software Audit Review**

  Audit means an independent inspection of a software product or processes by an external personnel to assess compliance with specifications, standards, contractual agreements, or other criteria. The aim of conducting software audit is to provide an independent evaluation of the software products and processes to applicable standards, guidelines, plans and procedures against compliance.

-(2.2.1)
-Software testing metrics - defect removal efficiency (DRE)

Defect Identification and Removal

A defect is a software bug that arises when the expected result doesn't match with the actual results. Most defects arise from mistakes and errors made by the developers and architects during the SDLC. The process of identifying these defects is known defect identification.

Methods and techniques like Peer Review, Code Analysis, Software Development methodologies and various kinds of programming techniques are adopted. A list of defects are prepared after the identification and then reviews are conducted to debug. The cost of removing the defects increases exponentially as we move along the SDLC.

Defect Detection Efficiency DDE is the percentage of defects defected during a phase/stage divided by the total number of defects. Or, it can also be defined as the percentage of defects detected prior to a phase/stage.

(2.2.2)
-https://www.google.com/amp/s/www.geeksforgeeks.org/formal-technical-review-ftr-in-software-engineering/amp/

-(2.2.3)(khush note)
-https://www.google.com/amp/s/www.geeksforgeeks.org/software-engineering-software-review/amp/

-(2.2.4)
Formal Design Review

Formal Design Review is a process where a design of the proposed system is evaluated against the requirements just to verify the outcomes of previous activities. The final product review triggers the product release. The objectives of the formal design review are given as follows:

• It evaluates the consistency, completeness and achievability of the proposed system according to the requirement.
• Physical tests can be performed according to the already set levels and milestones.
• It provides a basis for the engineering simulation.
• Walk-through is conducted just to ensure that the product is developed according to the user requirement or not.
• In order to develop large systems it examines the allocation of resources to individual configuration items.
• These designs are developed for technical adequacy and feasibility.
• It also provides the relative comparisons for merits and demerits of the system developed so far.

-(2.2.5)
-Code Review

Unit-3

-(3.1)
-Standards and Certificates

Quality Assurance Standards

Quality assurance standards are defined as frameworks that provide regulations to organizations to ensure that their processes, inputs, products, and services are capable of meeting every customer requirement. Organizations chiefly seek regulatory standards and frameworks that define specific practices because they help them achieve many objectives. The key objectives are:

- Ensuring maximum satisfaction of clients by meeting their quality requirements
- Safety of products and services during use
- Complying with international regulations and local legislative rules
- Being environmentally responsible
- Confidentiality of stakeholders including customers, employees, partners, and investors.
- Assuring a safer workspace for employees
- Optimum allocation of resources and minimization of waste

The following institutes and organizations are the main developers of SQA and software engineering standards –

- IEEE (Institute of Electrical and Electronics Engineers) Computer Society
- ISO (International Organization for Standardization)
- DOD (US Department of Defense)
- ANSI (American National Standards Institute)
- IEC (International ElectroTechnical Commission)
- EIA (Electronic Industries Association)

SQA Standards

Software quality assurance standards can be classified into two main classes –

- Software quality assurance management standards, including certification and assessment methodologies (quality management standards)

- Software project development process standards (project process standards)

Quality Management Standards

These focus on the organization's SQA system, infrastructure and requirements, while leaving the choice of methods and tools to the organization. With quality management standards, organizations can steadily assure that their software products achieve an acceptable level of quality.

Example – ISO 9000-3 and the Capability Maturity Model (CMM)

Project Process Standards

These focus on the methodologies for implementing the software development and maintenance projects. These standards include the following –

- The steps to be taken
- Design documentation requirements
- Contents of design documents
- Design reviews and review issues
- Software testing to be performed
- Testing topics

Example - ISO/IEC 12207, IEEE Std 1012 1998

-(3.2)
(ISO 9000)
ISO 9000 17somya

ISO 9000 Standards

The International Organization for Standardization ISO is an independent, non-governmental organization made up of members from the national bodies of over 160 countries that set international standards related to products and services.

The ISO 9000 series of standards, related to quality management, is perhaps the most widely known and impactful of any standards issues by ISO. ISO 9000 is a family of internationally accepted standards that are used to measure the quality management system followed by an organization.

The main purpose of the ISO 9000 standards is to provide a time-tested framework to help companies establish and follow a systematic approach for managing organizational processes for rendering consistent quality.

ISO 9000 is often used to refer to a family of three standards:

- ISO 9000 Fundamentals and vocabulary
- ISO 9001 - Requirements
- ISO 9004 - Guidelines for performance improvement

ISO 9000 explains the principles of the quality management system while ISO 9001 defines the requirements that an organization has to meet to gain certification.

ISO 9000 contains the various definitions and terminologies that are integral to developing a proper understanding of the quality management concepts used by ISO 9001.
The main advantages of ISO 9000 include the creation and continual improvement of effective and efficient operating processes, reducing waste, increasing productivity, better marketing and above all else - increasing customer satisfaction and retention.

ISO 9000 principles

- ensuring a focus on customer satisfaction
- developing leadership that drives the purpose of the organization
- adopting the process approach to managing activities and resources
- applying a system based approach to manage interrelated processes
- continual improvement
- using data to drive decision making

(ISO 9001:2000)
ISO 9001:2000 - Quality management systems — Requirements

ISO 9001: 2000

The **ISO 9001 :2000** specified requirements for a quality management system where an organization:

- needs to demonstrate its ability to consistently provide product

  that meet customer and applicable regulatory requirements, and

- aims to enhance customer satisfaction through effective application of the system, including processes for continual improvement of the system and the assurance of conformity to customer and applicable regulatory requirements.

(ISO 9001:2008)
ISO 9001:2008 - Quality management systems — Requirements

ISO 9001 2008

The **ISO 9001 2008** revision introduces a concept known as the process model. This means that you need to define what an organization does by:

- developing a process model of the organization's activities
- understanding how those processes inter-relate
- deciding who owns these processes and ensure they are trained and competent

- monitoring and improving the system by auditing and measuring customer satisfaction.

(CMM)
https://www.google.com/amp/s/www.geeksforgeeks.org/software-engineering-capability-maturity-model-cmm/amp/

(CMMi)
Capability Maturity Model Integration CMMi In Software Testing

Unit-4

-(4.1) Metrics & Measurements and Measurement Principals
+
(4.2) Attributes of Effective Software Metrics(**Characteristics)**
https://www.google.com/amp/s/www.geeksforgeeks.org/software-measurement-and-metrics/amp/

-(4.3)
https://www.google.com/amp/s/www.simplilearn.com/project-and-process-metrics-article/amp

-(4.4)
https://www.1000sourcecodes.com/2012/05/software-engineering-metrics-for_161.html?m=1

⬆️ YouTube..

-(4.5)

-(4.6)

-(4.7)
https://www.javatpoint.com/software-engineering-software-reliability-measurement-techniques