Software quality measures how well software is designed (*quality of design*), and how well the software conforms to that design (*quality of conformance*),Whereas *quality of conformance* is concerned with implementation , *quality of design* measures how valid the design and requirements are in creating a worthwhile product.

Objectives/Importance/Goals of Software Quality :-

Predictability

Software quality drives predictability. Do it once and do it right, and there will be less re-work, less variation in productivity and better performance overall. Products get delivered on time, and they get built more productively. Poor quality is much more difficult to manage. Predictability decreases as re-work grows, and the likelihood of a late, lower quality product increases.

Reputation

Some companies have a reputation for building quality software. It becomes who they are, a part of their brand, and ultimately it is the expectation people have of them. Customers seek them out because of it. A good, solid reputation is hard to establish and easy to lose, but when your company has it, it's a powerful business driver. A few mistakes and that reputation can be gone, creating major obstacles to sales, and consequently, your bottom line.

Employee Morale

The most productive and happy employees have pride in their work. Enabling employees to build quality software will drive a much higher level of morale and productivity. On the other hand, poor products, lots of re-work, unhappy customers and difficulty making deadlines have the opposite effect, leading to expensive turnover and a less productive workforce.

Customer Satisfaction

A quality product satisfies the customer. A satisfied customer comes back for more and provides positive referrals. Customer loyalty is heavily driven by the quality of the software you produce and service you provide. And, let's not forget that with the explosion of social media channels such as Twitter and Facebook, positive referrals can spread quickly. Of course that means poor quality and dissatisfaction can also be communicated quickly, if not even quicker than the good ones.

Bottom Line

It all drives the bottom line. Predictable and productive performance, a stellar reputation, happy employees and satisfied customers are the formula for a successful software business.

Few other stand-alone elements of a successful business have as deep an impact as Software Quality.

Software Quality Attributes are: Correctness, Reliability, Adequacy, Learnability, Robustness, Maintainability, Readability, Extensibility, Testability, Efficiency, Portability.

1. **Correctness:** The correctness of a software system refers to:

- Agreement of program code with specifications
- Independence of the actual application of the software system.

The correctness of a program becomes especially critical when it is embedded in a complex software system.

2. **Reliability:** Reliability of a software system derives from

- Correctness
- Availability

The behavior over time for the fulfillment of a given specification depends on the reliability of the software system.

3. **Reliability** of a software system is defined as the probability that this system fulfills a function (determined by the specifications) for a specified number of input trials under specified input conditions in a specified time interval (assuming that hardware and input are free of errors).

A software system can be seen as reliable if this test produces a low error rate (i.e., the probability that an error will occur in a specified time interval.)

The error rate depends on the frequency of inputs and on the probability that an individual input will lead to an error.

4. **Adequacy: Factors for the requirement of Adequacy:**

- The input required of the user should be limited to only what is necessary. The software system should expect information only if it is necessary for the functions that the user wishes to carry out. The software system should enable flexible data input on the part of the user and should carry out plausibility checks on the input. In dialog-driven software systems, we vest particular importance in the uniformity, clarity and simplicity of the dialogs.

- The performance offered by the software system should be adapted to the wishes of the user with the consideration given to extensibility; i.e., the functions should be limited to these in the specification.

5. **The results produced by the software system:** The results that a software system delivers should be output in a clear and wellstructured form and be easy to interpret. The software system should afford the user flexibility with respect to the scope, the degree of detail, and the form of presentation of the results. Error messages must be provided in a form that is comprehensible for the user.
6. **Learnability:** Learnability of a software system depends on:

- The design of user interfaces
- The clarity and the simplicity of the user instructions (tutorial or user manual).

The user interface should present information as close to reality as possible and permit efficient utilization of the software's failures.

The user manual should be structured clearly and simply and be free of all dead weight. It should explain to the user what the software system should do, how the individual functions are activated, what relationships exist between functions, and which exceptions might arise and how they can be corrected. In addition, the user manual should serve as a reference that supports the user in quickly and comfortably finding the correct answers to questions.

7. **Robustness:** Robustness reduces the impact of operational mistakes, erroneous input data, and hardware errors.

Software quality factors

Some software quality factors are listed here:

1. **Understandability**
   Clarity of purpose. ; All of the design and user documentation must be clearly written so that it is easily understandable. for example, if the software product is to be used by software

engineers it is not required to be understandable to the layman.so proper documentation should exist for layman's understanding.

2. Completeness

All required input data must also be available.All software tools in userinterface must be in fully complete form.

3.Conciseness

Minimization of excessive or redundant information or processing. This is important where memory capacity is limited, and it is generally considered good practice to keep lines of code to a minimum. Duplicacy must be avoided in case of coding as well as in documentation.

4.Portability

Ability to be run well and easily on multiple computer configurations. Portability can mean both between different hardware—such as running on a PC as well as a smartphone—and between different operating systems—such as running on both Mac OS X and GNU/Linux.

5.Consistency

Uniformity in notation, symbology, appearance, and terminology within itself.

6.Maintainability

Propensity to facilitate updates to satisfy new requirements. Thus the software product that is maintainable should be well-documented, should not be complex, and should have spare capacity for memory, storage and processor utilization and other resources.

7.Testability

Under software evaluation,software should be able to test using any tseting criteria . Such a characteristic must be built-in during the design phase if the product is to be easily testable; a complex design leads to poor testability.

8.Usability

User Interface should be user friendly. This is affected by such things as the human-computer interface. The component of the software that has most impact on this is the user interface (UI), which for best usability is usually graphical (i.e. a GUI).

9.Reliability

Ability to be expected to perform its intended functions satisfactorily. This implies a time factor in that a reliable product is expected to perform correctly over a period of time. It also encompasses environmental considerations in that the product is required to perform correctly in whatever conditions it finds itself (sometimes termed robustness).

10.Efficiency

Fulfillment of purpose without waste of resources, such as memory, space and processor utilization, network bandwidth, time, etc.

11.Security

Ability to protect data against unauthorized access and to withstand malicious or inadvertent interference with its operations. Besides the presence of appropriate security mechanisms such as authentication, access control and encryption, security also implies resilience in the face of malicious, intelligent and adaptive attackers.

What is Quality Assurance?

Quality Assurance is defined as the auditing and reporting procedures used to provide the stakeholders with data needed to make well-informed decisions.

It is the Degree to which a system meets specified requirements and customer expectations. It is also monitoring the processes and products throughout the SDLC.

Quality Assurance Criteria:

Below are the Quality assurance criteria against which the software would be evaluated against: 1.correctness 2. efficiency 3.flexibility 4.integrity 5.interoperability 6.maintainability 7.portability 8.reliability 9.reusability 10.testability 11.usability

SQA Activities: / Steps / Methods / Procedures / Function

- Software quality assurance(SQA) is composed of a variety of tasks associated with two different constituencies – the software engineers who do technical work and an SQA group that has responsibility for quality assurance planning, record keeping, analysis and reporting.Following activities are performed by an independent SQA group:

1. **Prepares an SQA plan for a project:** The plan is developed during project planning and is reviewed by all stakeholders. Quality assurance activities performed by the software engineering team and the SQA group are governed by the plan. The plan identifies evalu`ations to be performed, audits and reviews to be performed, standards that are applicable to the project, procedures for error reporting and tracking, documents to be produced by the SQA team, and amount of feedback provided to the software project team.
2. **Participates in the development of the project's software process description:** The software team selects a process for the work to be performed. The SQA group reviews the process description for compliance with organizational policy, internal software standards, externally imposed standards (eg. ISO-9001), and other parts of software project plan.

3. **Reviews software engineering activities to verify compliance with the defined software process:** The SQA group identifies, documents, and tracks deviations from the process and verifies that corrections have been made.

4. **Audits designated software work products to verify compliance with those defined as a part of the software process :** The SQA group reviews selected work products, identifies, documents and tracks deviations, verifies that corrections have been made, and periodically reports the results of its work to the project manager.

5. **Ensures that deviations in software work and work products are documented and handled according to a documented procedure:** Deviations may be encountered in the project plan, process description, applicable standards, or technical work products.

6. **Records any noncompliance and reports to senior management:** Non- compliance items are tracked until they are resolved.

Goals and Objectives of SQA:

Software Quality Assurance was created with the following objectives:

1. **Error Free (Small to Zero Defects After Installation)** :- The biggest goals of Software Quality Assurance is to prevent any possible defects when the output is made.The ability to handle stress of a program is different from the errors it has but crashes usually comes from defects so prevention of defects will most likely yield a continuously working application.
2. **Customer Satisfaction** :- Part of Software Quality Assurance is to ensure that software development was made according to their needs, wants and exceeding their expectations. Even if the bugs and errors are minimized by the system, customer satisfaction is more important and should be emphasized.
3. **Well Structured** :- Software Quality Assurance ensures that each application are build in an understandable manner. Their applications could easily be transferred from one developer to another.

What is Quality Control?

Quality control is a set of methods used by organizations to achieve quality parameters or quality goals and continually improve the organization's ability to ensure that a software product will meet quality goals.

Quality control provides monitoring the software development process to ensure that quality assurance procedures and standards are being followed. The deliverables from the software development process are checked against the defined project standards in the quality control process. The quality of software project deliverables can be checked by regular quality reviews and/or automated software assessment. Quality reviews are performed by a group of people. They review the software and software process in order to check that the project standards have been followed and that software and documents conform to these standards. Automated software assessment processes the software by a program that compares it to the standards applied to the development project.

A) Quality reviews

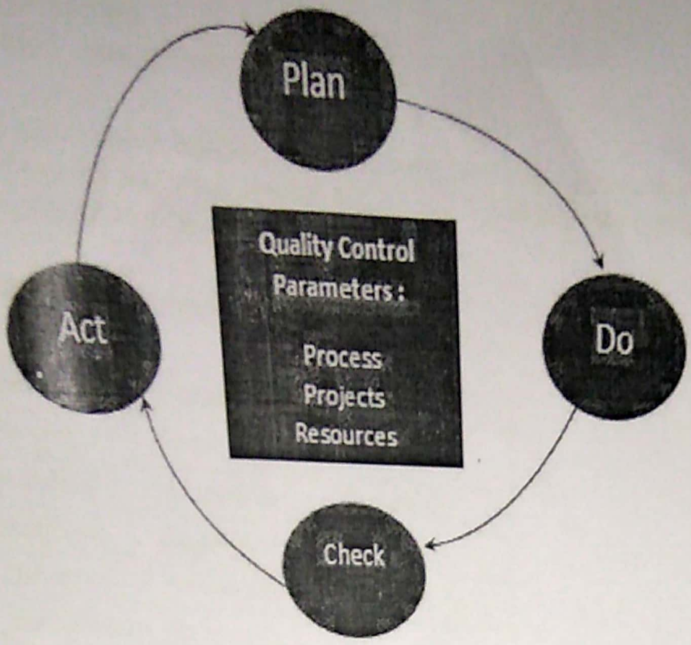
Quality reviews are the most widely used method of validating the quality of a process or product. They involve a group of people examining part or all of a software process, system, or its associated documentation to discover potential problems. The conclusions of the review are formally recorded and passed to the author for correcting the discovered problems. Table 1. describes several types of review, including quality reviews.

Table 1. Types of review.

Review type	Principal purpose
Design or program inspections	To detect detailed errors in the requirements, design or code.
Progress reviews	To provide information for management about the overall progress of the project.
Quality reviews	To carry out a technical analysis of product components or documentation to find mismatches between the specification and the component design, code or documentation and to ensure that defined quality standards of the organization have been followed.

Quality Control Process:

Quality Control process / steps



The three class parameters that control software quality are:

1.Products

2.Processes

3.Resources

The total quality control process consists of:

Plan - It is the stage where the Quality control processes are planned

Do - Use a defined parameter to develop the quality

Check - Stage to verify if the quality of the parameters are met

Act - Take corrective action if needed and repeat the work

Quality Control characteristics:

Process adopted to deliver a quality product to the clients at best cost.

Goal is to learn from other organizations so that quality would be better each time.

To avoid making errors by proper planning and execution with correct review process.

Quality planning

Quality planning is the process of developing a quality plan for a project. The quality plan defines the quality requirements of software and describes how these are to be assessed. The quality plan selects those organizational standards that are appropriate to a particular product and development process. Quality plan has the following parts:

1. Introduction of product.
2. Product plans.

3. Process descriptions.
4. Quality goals.
5. Risks management.

The quality plan defines the most important quality attributes for the software and includes a definition of the quality assessment process. Table 1 shows generally used software quality attributes that can be considered during the quality planning process.

Table 1.Software quality attributes.

Safety	Understandability	Portability
Security	Testability	Usability
Reliability	Adaptability	Reusability
Resilience	Modularity	Efficiency
Robustness	Complexity	Learnability
Maintainability	聽	

Software Quality Standards

Many organizations around the globe develop and implement different standards to improve the quality needs of their software. This chapter briefly describes some of the widely used standards related to Quality Assurance and Testing.

ISO/IEC 9126

This standard deals with the following aspects to determine the quality of a software application:

- Quality model
- External metrics
- Internal metrics
- Quality in use metrics

This standard presents some set of quality attributes for any software such as:

- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability
- Portability

The above-mentioned quality attributes are further divided into sub-factors, which you can study when you study the standard in detail.

ISO/IEC 9241-11

Part 11 of this standard deals with 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.

This standard proposed a framework that describes the usability components and the relationship between them. In this standard, the usability is considered in terms of user performance and satisfaction.

According to ISO 9241-11, usability depends on context of use and the level of usability will change as the context changes.

ISO/IEC 25000:2005

ISO/IEC 25000:2005 is commonly known as the standard that provides the guidelines for Software Quality Requirements and Evaluation (SQuaRE). This standard helps in organizing and enhancing the process related to software quality requirements and their evaluations. In reality, ISO-25000 replaces the two old ISO standards, i.e. ISO-9126 and ISO-14598.

SQuaRE is divided into sub-parts such as:

- ISO 2500n - Quality Management Division
- ISO 2501n - Quality Model Division
- ISO 2502n - Quality Measurement Division
- ISO 2503n - Quality Requirements Division
- ISO 2504n - Quality Evaluation Division

The main contents of SQuaRE are:

- Terms and definitions
- Reference Models
- General guide
- Individual division guides
- Standard related to Requirement Engineering (i.e. specification, planning, measurement and evaluation process)

ISO/IEC 12119

This standard deals with software packages delivered to the client. It does not focus or deal with the clients' production process. The main contents are related to the following items:

- Set of requirements for software packages.
- Instructions for testing a delivered software package against the specified requirements.

Miscellaneous Standards

Some of the other standards related to QA and Testing processes are mentioned below:

Standard	Description
IEEE 829	A standard for the format of documents used in different stages of software testing.
IEEE 1061	A methodology for establishing quality requirements, identifying, implementing, analyzing, and validating the process, and product of software quality metrics.
IEEE 1059	Guide for Software Verification and Validation Plans.
IEEE 1008	A standard for unit testing.
IEEE 1012	A standard for Software Verification and Validation.
IEEE 1028	A standard for software inspections.
IEEE 1044	A standard for the classification of software anomalies.
IEEE 1044-1	A guide for the classification of software anomalies.
IEEE 830	A guide for developing system requirements specifications.
IEEE 730	A standard for software quality assurance plans.
IEEE 1061	A standard for software quality metrics and methodology.
IEEE 12207	A standard for software life cycle processes and life cycle data.

BS 7925-1	A vocabulary of terms used in software testing.
BS 7925-2	A standard for software component testing.