8.1 Software quality attributes

Software Quality Attributes are features that facilitate the measurement of performance of a software product by Software Testing professionals, and include attributes such as availability, interoperability, correctness, reliability, robustness, maintainability, readability, extensibility, testability, efficiency, and portability.

High scores in Software Quality Attributes enable software architects to guarantee that a software application will perform as the specifications provided by the client.

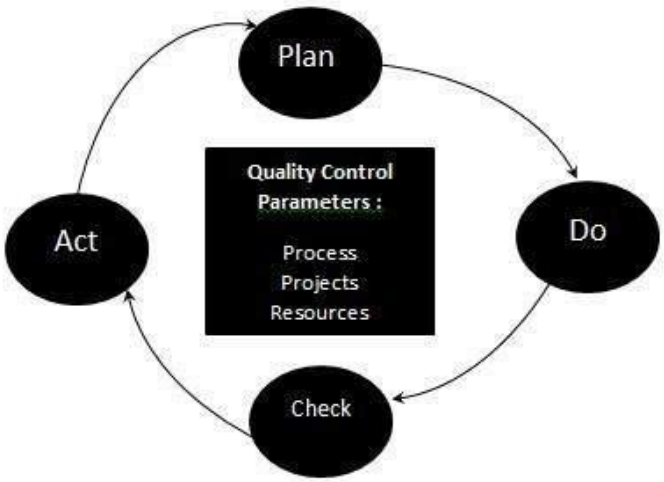
8.2 Quality factors

The various factors, which influence the software, are termed as software factors. They can be broadly divided into two categories. The first category of the factors is of those that can be measured directly such as the number of logical errors, and the second category clubs those factors which can be measured only indirectly. For example, maintainability but each of the factors is to be measured to check for the content and the quality control.

8.3 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 Process:



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.

8.4 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:

- correctness
- efficiency
- flexibility
- integrity
- interoperability
- maintainability
- portability
- reliability
- reusability
- testability
- usability

Software quality assurance

Software Quality Assurance (SQA) is simply a way to assure quality in the software. It is the set of activities which ensure processes, procedures as well as standards are suitable for the project and implemented correctly.

Software Quality Assurance is a process which works parallel to development of software. It focuses on improving the process of development of software so that problems can be prevented before they become a major issue. Software Quality Assurance is a kind of Umbrella activity that is applied throughout the software process.

Software Quality Assurance has:

- A quality management approach
- Formal technical reviews
- Multi testing strategy
- Measurement and reporting mechanism

Benefits of Software Quality Assurance (SQA):

- SQA produces high quality software.
- High quality application saves time and cost.
- SQA is beneficial for better reliability.
- SQA is beneficial in the condition of no maintenance for a long time.

8.5 Verification and Validation

Validation refers to the set of authorities which ensure that software is correctly implemented for a specific function.

Verification refers to a different set of authorities which ensure that software which has been built is traceable to customer requirements.

Verification: - are we making the product?

Validation: - are we making the right product.

It is basically a part of the software quantity assurance (SQA): - tested approach in the S/E-

It includes the following objectivity-

1) To ensure that product is safe.

2) To ensure that product will function properly under both normal exceptional conditions.

3) To ensure that the product is what the user want.

8.6 Testing and Debugging

Testing is the process of executing a program or system with the intent of finding errors, verifying that it meets specified requirements, and ensuring that it behaves as expected. The goal of testing is to identify defects or bugs in the software and make sure it functions correctly. Testing is typically performed in different stages throughout the software development life cycle (SDLC), including unit testing, integration testing, system testing, and acceptance testing.

Types of testing include:

1. **Unit Testing**: It involves testing individual components or units of code to verify their correctness and functionality in isolation. Developers usually write unit tests to ensure that each unit performs as expected.
2. **Integration Testing**: It tests the interaction and communication between different components or modules to ensure they work together correctly.
3. **System Testing**: It verifies the behavior and functionality of a complete and integrated system. It tests the system as a whole to ensure that it meets the specified requirements.
4. **Acceptance Testing**: It involves testing the system with real-world scenarios and data to determine whether it satisfies the business requirements and is ready for deployment.
5. **Performance Testing**: It checks the performance and scalability of the software under various workloads and stress conditions to ensure it can handle the expected user load.

Debugging, on the other hand, is the process of identifying, analyzing, and fixing defects or bugs found during testing or in the production environment. Debugging is focused on understanding the root cause of a software issue and

making the necessary changes to resolve it. It often involves step-by-step execution of the program, examining variable values, and analyzing program flow to locate and fix the problem.

Debugging techniques and tools include:

1. **Logging**: Adding log statements in the code to capture information about the program's execution, variable values, and error conditions.
2. **Debugging Tools**: Integrated development environments (IDEs) provide debugging tools that allow developers to set breakpoints, step through code, inspect variables, and track program flow.
3. **Code Review**: Collaborating with peers to review code and identify potential issues or bugs.
4. **Error Analysis**: Analyzing error messages, stack traces, and crash dumps to understand the cause of the problem.

Testing and debugging are essential parts of the software development process. They help identify and fix issues early, leading to more robust and reliable software applications.

8.7 Testing Process

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. In simple words, testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

According to ANSI/IEEE 1059 standard, Testing can be defined as - A process of analyzing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item.

General characteristics of strategic testing

- To perform effective testing, a s/w team should conduct technical reviews
- Testing begins at the component level and work toward the integration of entire computer based system.

- Different testing techniques are appropriate at different points in time.
- Testing is done by developer of s/w and for large projects by an independent test group.
- Testing and debugging are different activities, but debugging must be accommodated in any test strategy.

In other words software testing is a verification and validation process

Verification: "Does the product meet its specification?" The set of activities that ensure the s/w correctly implements a specific function or algorithm.

Validation: "Does the product perform as desired?" The set of activities that ensure that the s/w has been built is traceable to customer requirement.

8.8 Unit testing

- Unit testing is the first level of functional testing in order to test any software. In this, the test engineer will test the module of an application independently or test all the module functionality is called unit testing.
- The primary objective of executing the unit testing is to confirm the unit components with their performance.
- Here, a unit is defined as a single testable function of software or an application. And it is verified throughout the specified application development phase.
- If some code which is dealing with the read and write function on database we cannot test that as a unit because the database is linked with it now.

8.9 Integration testing

Upon completion of unit testing, the units or modules are to be integrated which gives raise to integration testing. The purpose of integration testing is to verify the functional, performance, and reliability between the modules that are integrated.

Integration Strategies:

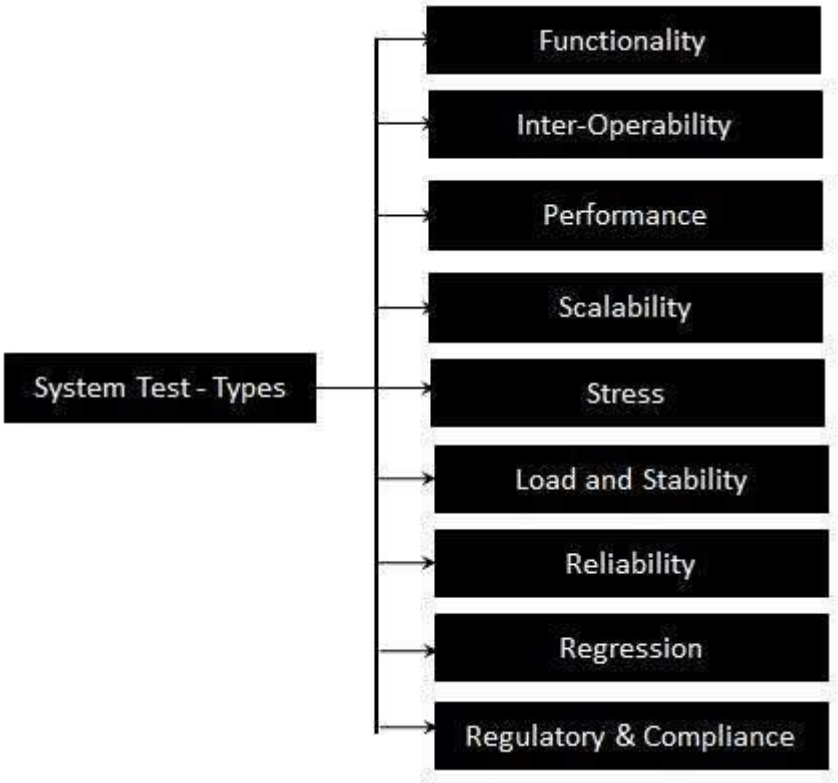
- Big-Bang Integration
- Top Down Integration
- Bottom Up Integration
- Hybrid Integration

8.10 System testing

System Testing (ST) is a black box testing technique performed to evaluate the complete system the system's compliance against specified requirements. In System testing, the functionalities of the system are tested from an end-to-end perspective.

System Testing is usually carried out by a team that is independent of the development team in order to measure the quality of the system unbiased. It includes both functional and Non-Functional testing.

Types of System Tests:



8.11 Regression testing

Regression testing a black box testing technique that consists of re-executing those tests that are impacted by the code changes. These tests should be executed as often as possible throughout the software development life cycle.

Types of Regression Tests:

Final Regression Tests: - A "final regression testing" is performed to validate the build that hasn't changed for a period of time. This build is deployed or shipped to customers.

Regression Tests: - A normal regression testing is performed to verify if the build has NOT broken any other parts of the application by the recent code changes for defect fixing or for enhancement.

Selecting Regression Tests:

- Requires knowledge about the system and how it affects by the existing functionalities.
- Tests are selected based on the area of frequent defects.
- Tests are selected to include the area, which has undergone code changes many times.
- Tests are selected based on the criticality of the features.

Regression Testing Steps:

Regression tests are the ideal cases of automation which results in better Return On Investment (ROI).

- Select the Tests for Regression.
- Choose the apt tool and automate the Regression Tests
- Verify applications with Checkpoints
- Manage Regression Tests/update when required
- Schedule the tests
- Integrate with the builds
- Analyze the results

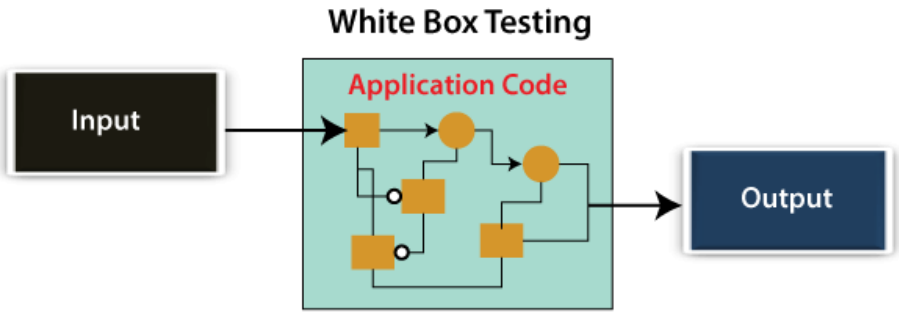
8.12 White box testing and Black box testing

In white-box testing, the developer will inspect every line of code before handing it over to the testing team or the concerned test engineers.

Subsequently, the code is noticeable for developers throughout testing; that's why this process is known as **WBT (White Box Testing)**.

In other words, we can say that the **developer** will execute the complete white-box testing for the particular software and send the specific application to the testing team.

The purpose of implementing the white box testing is to emphasize the flow of inputs and outputs over the software and enhance the security of an application.



White box testing is also known as **open box testing, glass box testing, structural testing, clear box testing, and transparent box testing**.

Black Box Testing

Another type of manual testing is **black-box testing**. In this testing, the test engineer will analyze the software against requirements, identify the defects or bug, and sends it back to the development team.

Then, the developers will fix those defects, do one round of White box testing, and send it to the testing team.

Here, fixing the bugs means the defect is resolved, and the particular feature is working according to the given requirement.

The main objective of implementing the black box testing is to specify the business needs or the customer's requirements.

In other words, we can say that black box testing is a process of checking the functionality of an application as per the customer requirement. The source code is not visible in this testing; that's why it is known as **black-box testing**.

Types of Black Box Testing

Black box testing further categorizes into two parts, which are as discussed below:

- **Functional Testing**
- **Non-function Testing**