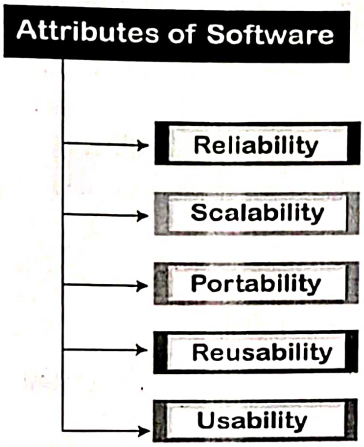
Unit :- 1 & 2

What is Software Testing

Software testing is a process of identifying the correctness of software by considering its all attributes (Reliability, Scalability, Portability, Re-usability, Usability) and evaluating the execution of software components to find the software bugs or errors or defects.

execution of s/w component to find s/w bugs or Errors or defects.



Software testing provides an independent view and objective of the software and gives surety of fitness of the software. It involves testing of all components under the required services to confirm that whether it is satisfying the specified requirements or not. The process is also providing the client with information about the quality of the software.

Testing is mandatory because it will be a dangerous situation if the software fails any of time due to lack of testing. So, without testing software cannot be deployed to the end user.

What is Testing

Testing is a group of techniques to determine the correctness of the application under the predefined script but, testing cannot find all the defect of application. The main intent of testing is to detect failures of the application so that failures can be discovered and corrected. It does not demonstrate that a product functions properly under all conditions but only that it is not working in some specific conditions.

Testing furnishes comparison that compares the behavior and state of software against mechanisms because the problem can be recognized by the mechanism. The mechanism may include past versions of the same specified product, comparable products, and interfaces of expected purpose, relevant standards, or other criteria but not limited up to these.

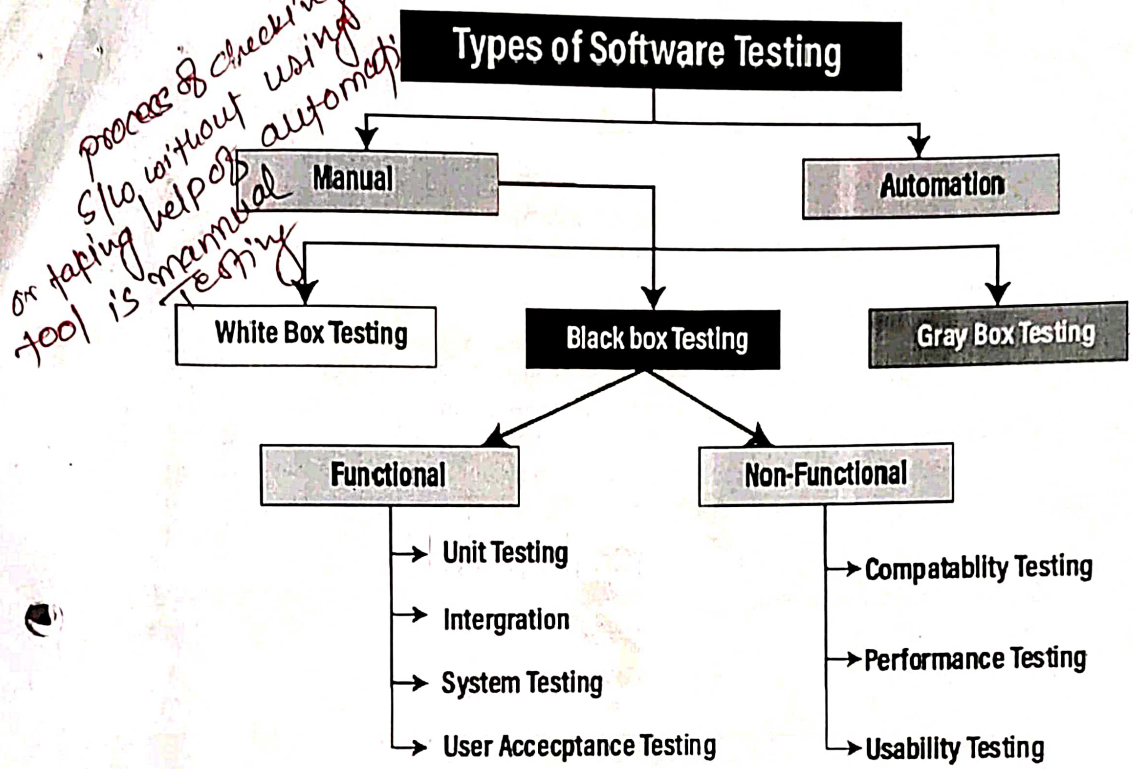
Testing includes an examination of code and also the execution of code in various environments, conditions as well as all the examining aspects of the code. In the current scenario of software development, a testing team may be separate from the development team so that Information derived from testing can be used to correct the process of software development.

The success of software depends upon acceptance of its targeted audience, easy graphical user interface, strong functionality load test, etc. For example, the audience of banking is totally different from the audience of a video game. Therefore, when an organization develops a software product, it can assess whether the software product will be beneficial to its purchasers and other audience.

Type of Software testing

We have various types of testing available in the market, which are used to test the application or the software.

With the help of below image, we can easily understand the type of software testing:



process of checking s/w without using automation or taping help of tool is manual testing

Manual testing

The process of checking the functionality of an application as per the customer needs without taking any help of automation tools is known as manual testing. While performing the manual testing on any application, we do not need any specific knowledge of any testing tool, rather than have a proper understanding of the product so we can easily prepare the test document.

Manual testing can be further divided into three types of testing, which are as follows:

- **White box testing**
- **Black box testing**
- **Gray box testing**

For more information about manual testing, refers to the below link:

Automation testing

Automation testing is a process of converting any manual test cases into the test scripts with the help of automation tools, or any programming language is known as automation testing. With the help of automation testing, we can enhance the speed of our test execution because here, we do not require any human efforts. We need to write a test script and execute those scripts.

enhance speed
write test script & execute test script

V-Model

The V-model of SDLC carries out its execution in a sequential manner. The structure it follows takes the shape of the letter V. This model is also popularly termed as a **Verification and Validation model**. Here, each phase has to be finished before beginning the next phase. A sequential design progression is followed like that of the waterfall model.

In parallel to the software development phase, a corresponding series of test phase also runs in this model. Each stage comprises a specific type of testing done, and once that testing is passed, only then the next phase starts.

1. **Verification:** In the concept of verification in the V-Model, static analysis technique is carried out without executing the code. This evaluation procedure is carried out at the time of development to check whether specific requirements will meet or not.
2. **Validation:** This concept of V-Model comprises of dynamic analysis practice (both functional as well as non-functional), and testing is done by code execution. The validation of a product is done once the development is complete for determining if the software meets up the customer hope needs.

So both verification and validation are combined and work in parallel to make the V-Model fully functional.

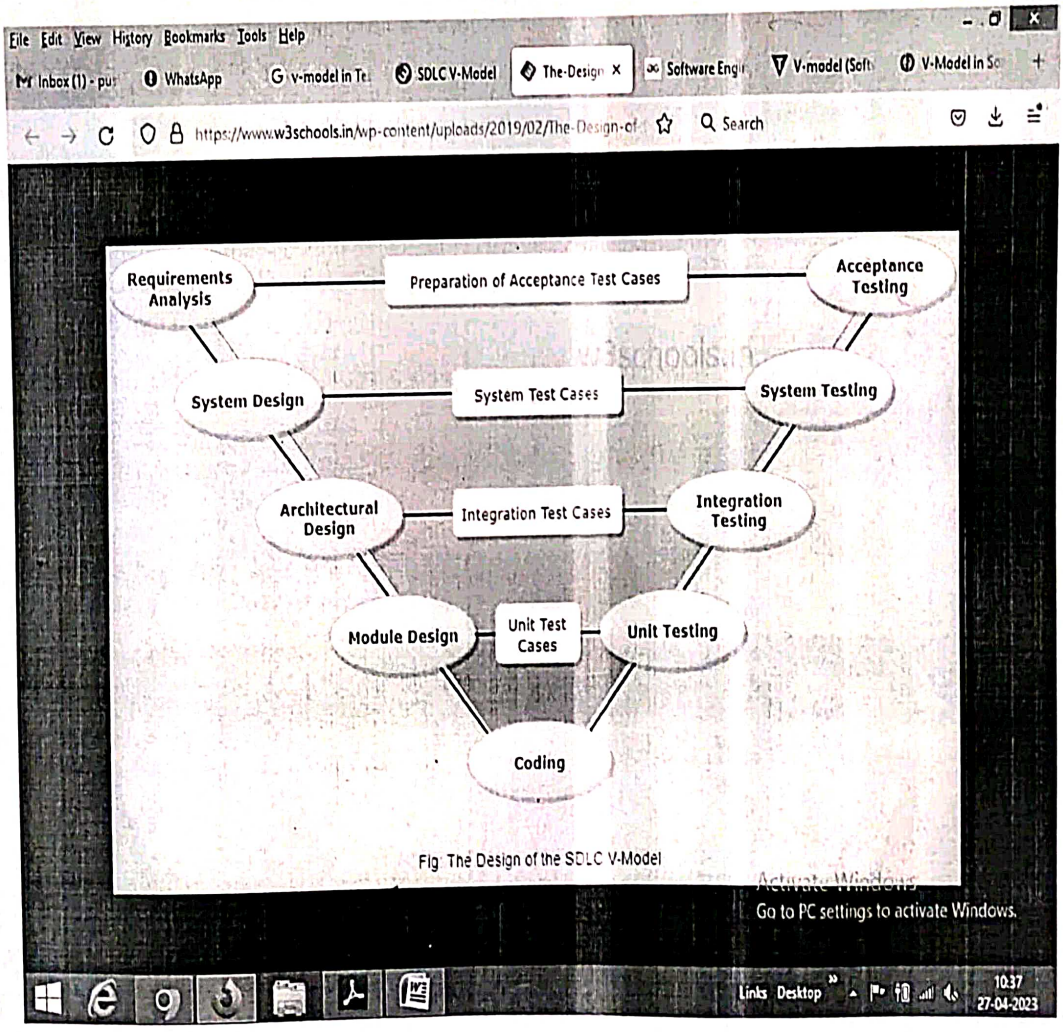


Fig: The Design of the SDLC V-Model

What is Test case?

A test case is a document, which has a set of test data, preconditions, expected results and post conditions, developed for a particular test scenario in order to verify compliance against a specific requirement.

Test Case acts as the starting point for the test execution, and after applying a set of input values, the application has a definitive outcome and leaves the system at some end point or also known as execution postcondition.

Typical Test Case Parameters:

- Test Case ID
- Test Scenario
- Test Case Description
- Test Steps
- Prerequisite
- Test Data
- Expected Result
- Test Parameters
- Actual Result
- Environment Information
- Comments

Example:

Let us say that we need to check an input field that can accept maximum of 10 characters.

While developing the test cases for the above scenario, the test cases are documented the following way. In the below example, the first case is a pass scenario while the second case is a FAIL.

Scenario	Test Step	Expected Result	Actual Outcome
Verify that the input field that can accept maximum of 10 characters	Login to application and key in 10 characters	Application should be able to accept all 10 characters.	Application accepts all 10 characters.
Verify that the input field that can accept maximum of 11 characters	Login to application and key in 11 characters	Application should NOT accept all 11 characters.	Application accepts all 10 characters.

If the expected result doesn't match with the actual result, then we log a defect. The defect goes through the defect life cycle and the testers address the same after fix.

Test case template

The primary purpose of writing a test case is to achieve the efficiency of the application.

Header

- Test Case Name/ID :-Release - Version - Application Name - Module
- Test Case Type:- F.T.C | I.T.C | S.T.C
- Requirement Number:-
- Module:-
- Serverity:- Critical/Major/Minor
- Status:-
- Release:-
- Version:-
- Pre-condition:-
- Test Data:-
- Summary:-

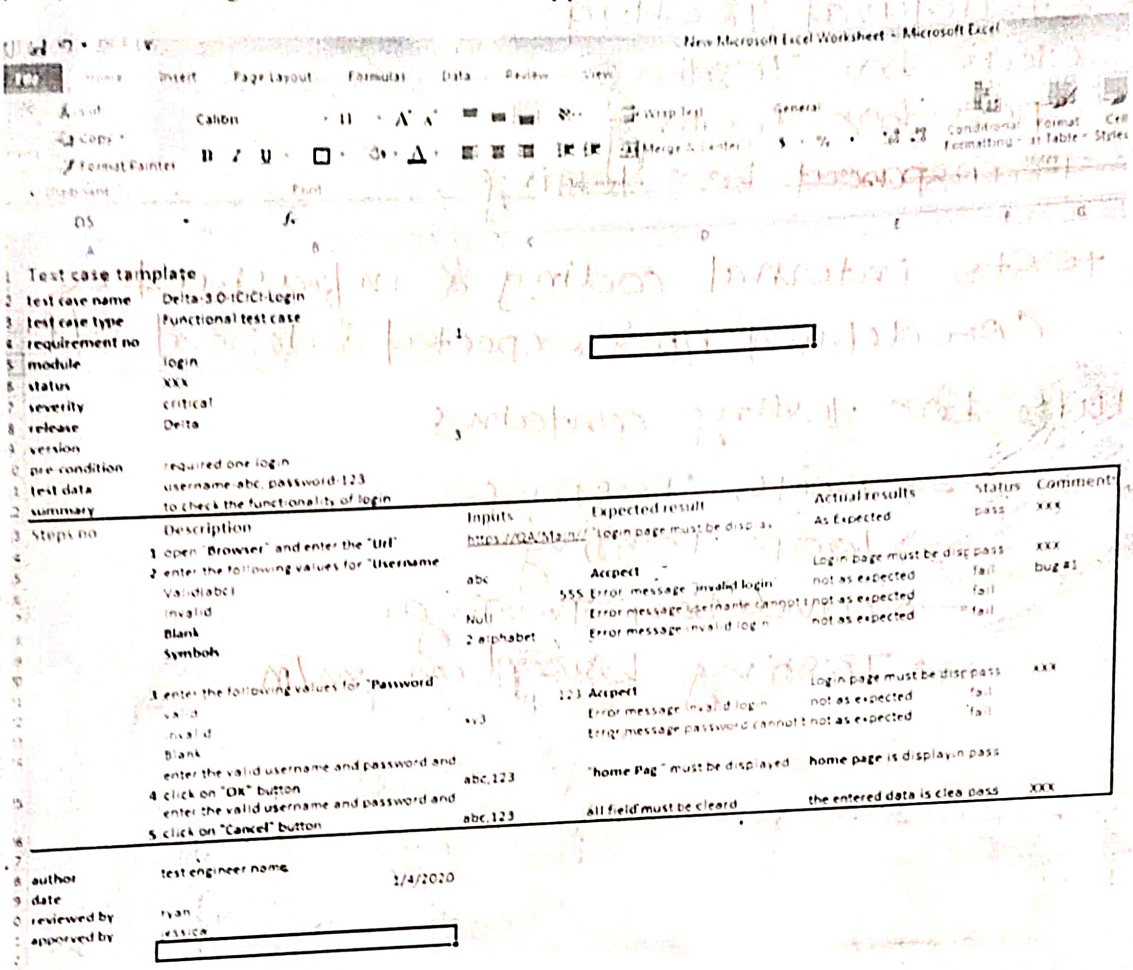
Body

Step No.	Descri-pation	Inputs	Expected Result	Actual Reasult	Status	Comments
...	...	...	...	...	...	...
...	...	...	...	...	...	...

Footer

- Author:- Reviewd By:-
- Date:- Approved By:-

Here, we are writing a test case for the **ICICI application's Login** module:



Test case tamplate						
test case name	Delta-3.0-ICICI-Login					
test case type	Functional test case					
requirement no						
module	login					
status	XXX					
severity	critical					
release	Delta					
version						
pre-condition	required one login					
test data	username:abc, password:123					
summary	to check the functionality of login					
Steps no	Description	Inputs	Expected result	Actual results	Status	Comments
	1 open "Browser" and enter the "Url"	https://qa.ma.in/	"Login page must be disp a	As Expected	pass	xxx
	2 enter the following values for "Username			Login page must be disp	pass	xxx
	Validlabel	abc	Accpect	not as expected	fail	bug #1
	Invalid		555 Error message "invalid login	not as expected	fail	
	Blank	Null	Error message username cannot t	not as expected	fail	
	Symboh	2 alphabet	Error message "invalid log n	not as expected	fail	
	3 enter the following values for "Password"			Login page must be disp	pass	xxx
	valid		123 Accpect	not as expected	fail	
	invalid	xx3	Error message "invalid login	not as expected	fail	
	Blank		Error message password cannot t			
	enter the valid username and password and					
	4 click on "OK" button	abc,123	"home Pag " must be displayed	home page is displayin	pass	
	enter the valid username and password and					
	5 click on "Cancel" button	abc,123	all field must be cleard	the entered data is clea	pass	xxx
author	test engineer name					
date		1/4/2020				
reviewed by	ryan					
apporved by	jessica					

1.Write a Test case for Website Login.

2. Write a Test case for College website Users Login.

3. Write a test case for Hospital Appointment for a patient.

White Box Testing :-

→ Structural Testing
Clear box Testing
Open box Testing
Transparent box Testing

→ tests internal coding & infrastructure
(predefined I/P & expected & desired O/P.

White box testing contains

→ path Testing
→ loop Testing
→ condition Testing
→ Testing based on m/m

9167370959.

White Box Testing

The box testing approach of software testing consists of black box testing and white box testing. We are discussing here white box testing which also known as glass box is **testing, structural testing, clear box testing, open box testing and transparent box testing.** It tests internal coding and infrastructure of a software focus on checking of predefined inputs against expected and desired outputs. It is based on inner workings of an application and revolves around internal structure testing. In this type of testing programming skills are required to design test cases. The primary goal of white box testing is to focus on the flow of inputs and outputs through the software and strengthening the security of the software.

① It tests internal coding & infrastructu

The term 'white box' is used because of the internal perspective of the system. The clear box or white box or transparent box name denote the ability to see through the software's outer shell into its inner workings.

Developers do white box testing. In this, the developer will test every line of the code of the program. The developers perform the White-box testing and then send the application or the software to the testing team, where they will perform the black box testing and verify the application along with the requirements and identify the bugs and sends it to the developer.

The developer fixes the bugs and does one round of white box testing and sends it to the testing team. Here, fixing the bugs implies that the bug is deleted, and the particular feature is working fine on the application.

Here, the test engineers will not include in fixing the defects for the following reasons:

- Fixing the bug might interrupt the other features. Therefore, the test engineer should always find the bugs, and developers should still be doing the bug fixes.
- If the test engineers spend most of the time fixing the defects, then they may be unable to find the other bugs in the application.

The white box testing contains various tests, which are as follows:

- Path testing
- Loop testing
- Condition testing
- Testing based on the memory perspective
- Test performance of the program

Generic steps of white box testing

- Design all test scenarios, test cases and prioritize them according to high priority number.
- This step involves the study of code at runtime to examine the resource utilization, not accessed areas of the code, time taken by various methods and operations and so on.
- In this step testing of internal subroutines takes place. Internal subroutines such as nonpublic methods, interfaces are able to handle all types of data appropriately or not.
- This step focuses on testing of control statements like loops and conditional statements to check the efficiency and accuracy for different data inputs.
- In the last step white box testing includes security testing to check all possible security loopholes by looking at how the code handles security.

Reasons for white box testing

- It identifies internal security holes.
- To check the way of input inside the code.

- Check the functionality of conditional loops.
- To test function, object, and statement at an individual level.

Advantages of White box testing

- White box testing optimizes code so hidden errors can be identified.
- Test cases of white box testing can be easily automated.
- This testing is more thorough than other testing approaches as it covers all code paths.
- It can be started in the SDLC phase even without GUI.

Disadvantages of White box testing

- White box testing is too much time consuming when it comes to large-scale programming applications.
- White box testing is much expensive and complex.
- It can lead to production error because it is not detailed by the developers.
- White box testing needs professional programmers who have a detailed knowledge and understanding of programming language and implementation.

Techniques Used in White Box Testing

Data Flow Testing	Data flow testing is a group of testing strategies that examines the control flow of programs in order to explore the sequence of variables according to the sequence of events.
Control Flow Testing	Control flow testing determines the execution order of statements or instructions of the program through a control structure. The control structure of a program is used to develop a test case for the program. In this technique, a particular part of a large program is selected by the tester to set the testing path. Test cases represented by the control graph of the program.
Branch Testing	Branch coverage technique is used to cover all branches of the control flow graph. It covers all the possible outcomes (true and false) of each condition of decision point at least once.
Statement Testing	Statement coverage technique is used to design white box test cases. This technique involves execution of all statements of the source code at least once. It is used to calculate the total number of executed statements in the source code, out of total statements present in the source code.
Decision Testing	This technique reports true and false outcomes of Boolean expressions. Whenever there is a possibility of two or more outcomes from the statements like do while statement, if statement and case statement (Control flow statements), it is considered as decision point because there are two outcomes either true or false.

Black Box Testing

Black Box Testing is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioral Testing.

The above Black-Box can be any software system you want to test. For Example, an operating system like Windows, a website like Google, a database like Oracle or even your own custom application. Under Black Box Testing, you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation. Consider the following video tutorial-

Black Box Testing Techniques

Following are the prominent Test Strategy amongst the many used in Black box Testing

- **Equivalence Class Testing:** It is used to minimize the number of possible test cases to an optimum level while maintains reasonable test coverage.
- **Boundary Value Testing:** Boundary value testing is focused on the values at boundaries. This technique determines whether a certain range of values are acceptable by the system or not. It is very useful in reducing the number of test cases. It is most suitable for the systems where an input is within certain ranges.
- **Decision Table Testing:** A decision table puts causes and their effects in a matrix. There is a unique combination in each column.

Types of Black Box Testing

There are many types of Black Box Testing but the following are the prominent ones –

- **Functional testing** – This black box testing type is related to the functional requirements of a system; it is done by software testers.
- **Non-functional testing** – This type of black box testing is not related to testing of specific functionality, but non-functional requirements such as performance, scalability, usability.
- **Regression testing** – Regression Testing is done after code fixes, upgrades or any other system maintenance to check the new code has not affected the existing code.

Advantages

- Tests are done from a user's point of view and will help in exposing discrepancies in the specifications.

- Tester need not know programming languages or how the software has been implemented.
- Tests can be conducted by a body independent from the developers, allowing for an objective perspective and the avoidance of developer-bias.
- Test cases can be designed as soon as the specifications are complete.

Disadvantages

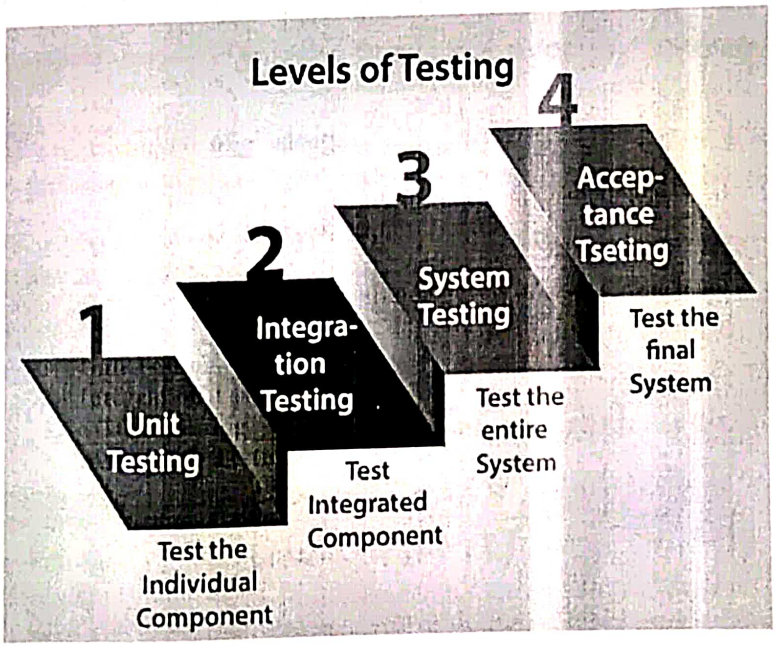
- Only a small number of possible inputs can be tested and many program paths will be left untested.
- Without clear specifications, which is the situation in many projects, test cases will be difficult to design.
- Tests can be redundant if the software designer/developer has already run a test case.
- Ever wondered why a soothsayer closes the eyes when foretelling events? So is almost the case in Black Box Testing.

Different Levels of Testing

The levels of software testing involve the different methodologies, which can be used while we are performing the software testing.

In software testing, we have four different levels of testing, which are as discussed below:

1. **Unit Testing**
2. **Integration Testing**
3. **System Testing**
4. **Acceptance Testing**



As we can see in the above image that all of these testing levels have a specific objective which specifies the value to the software development lifecycle.

For our better understanding, let's see them one by one:

Level1: Unit Testing

Unit testing is the first level of software testing, which is used to test if software modules are satisfying the given requirement or not.

The first level of testing involves **analyzing each unit or an individual component** of the software application.

Unit testing is also the first level of **functional testing**. The primary purpose of executing unit testing is to validate unit components with their performance.

A unit component is an individual function or regulation of the application, or we can say that it is the smallest testable part of the software. The reason of performing the unit testing is to test the correctness of inaccessible code.

Unit testing will help the test engineer and developers in order to understand the base of code that makes them able to change defect causing code quickly. The developers implement the unit.

For more information on unit testing, refers to the following link:

https://www.javatpoint.com/unit-testing.

Level2: Integration Testing

The second level of software testing is the **integration testing.** The integration testing process comes after **unit testing**.

It is mainly used to test the **data flow from one module or component to other modules.**

In integration testing, the **test engineer** tests the units or separate components or modules of the software in a group.

The primary purpose of executing the integration testing is to identify the defects at the interaction between integrated components or units.

When each component or module works separately, we need to check the data flow between the dependent modules, and this process is known as **integration testing**.

We only go for the integration testing when the functional testing has been completed successfully on each application module.

In simple words, we can say that **integration testing** aims to evaluate the accuracy of communication among all the modules.

For more information on integration testing, refers to the following link:

https://www.javatpoint.com/integration-testing.

Level3: System Testing

The third level of software testing is **system testing**, which is used to test the software's functional and non-functional requirements.

It is **end-to-end testing** where the testing environment is parallel to the production environment. In the third level of software testing, **we will test the application as a whole system.**

To check the end-to-end flow of an application or the software as a user is known as **System testing.**

In system testing, we will go through all the necessary modules of an application and test if the end features or the end business works fine, and test the product as a complete system.

In simple words, we can say that System testing is a sequence of different types of tests to implement and examine the entire working of an integrated software computer system against requirements.

Level4: Acceptance Testing

The **last and fourth level** of software testing is **acceptance testing**, which is used to evaluate whether a specification or the requirements are met as per its delivery.

The software has passed through three testing levels (**Unit Testing, Integration Testing, System Testing**). Some minor errors can still be identified when the end-user uses the system in the actual scenario.

In simple words, we can say that Acceptance testing is the **squeezing of all the testing processes that are previously done.**

The acceptance testing is also known as **User acceptance testing (UAT)** and is done by the customer before accepting the final product.

Usually, UAT is done by the domain expert (customer) for their satisfaction and checks whether the application is working according to given business scenarios and real-time scenarios.

What is Grey Box Testing? Techniques, Example

Grey Box Testing

Grey Box Testing or Gray box testing is a software testing technique to test a software product or application with partial knowledge of internal structure of the application. The purpose of grey box testing is to search and identify the defects due to improper code structure or improper use of applications.

In this process, context-specific errors that are related to web systems are commonly identified. It increases the testing coverage by concentrating on all of the layers of any complex system.

Gray Box Testing is a software testing method, which is a combination of both White Box Testing and Black Box Testing method.

- In White Box testing internal structure (code) is known
- In Black Box testing internal structure (code) is unknown
- In Grey Box Testing internal structure (code) is partially known

In Software Engineering, Gray Box Testing gives the ability to test both sides of an application, presentation layer as well as the code part. It is primarily useful in Integration Testing and ~~Penetration Testing~~.

Example of Gray Box Testing: While testing websites feature like links or orphan links, if tester encounters any problem with these links, then he can make the changes straightaway in HTML code and can check in real time.

Why Gray Box Testing

Gray Box Testing is performed for the following reason,

1. It provides combined benefits of both black box testing and white box testing both
2. It combines the input of developers as well as testers and improves overall product quality
   - It reduces the overhead of long process of testing functional and non-functional types
   - It gives enough free time for a developer to fix defects
   - Testing is done from the user point of view rather than a designer point of view

Gray Box Testing Strategy

To perform Gray box testing, it is not necessary that the tester has the access to the source code. A test is designed based on the knowledge of algorithm, architectures, internal states, or other high -level descriptions of the program behavior.

To perform Gray box Testing-

- It applies a straightforward technique of black box testing
- It is based on requirement test case generation, as such, it presets all the conditions before the program is tested by assertion method.

Techniques used for Grey box Testing are-

- **Matrix Testing:** This testing technique involves defining all the variables that exist in their programs.
- **Regression Testing**: To check whether the change in the previous version has regressed other aspects of the program in the new version. It will be done by testing strategies like retest all, retest risky use cases, retest within a firewall.
- **Orthogonal Array Testing or OAT**: It provides maximum code coverage with minimum test cases.
- **Pattern Testing:** This testing is performed on the historical data of the previous system defects. Unlike black box testing, gray box testing digs within the code and determines why the failure happened

Summary:

- The overall cost of system defects can be reduced and prevented from passing further with Grey box testing
- Grey box testing is suited more for GUI, Functional Testing, security assessment, web applications, web-services, etc.
- Techniques used for Grey box Testing
  - Matrix Testing
  - Regression Testing
  - OAT or Orthogonal Array Testing
  - Pattern Testing

Differences between Black Box Testing vs White Box Testing:

Black Box Testing	White Box Testing
It is a way of software testing in which the internal structure or the program or the code is hidden and nothing is known about it.	It is a way of testing the software in which the tester has knowledge about the internal structure or the code or the program of the software.
Implementation of code is not needed for black box testing.	Code implementation is necessary for white box testing.
It is mostly done by software testers.	It is mostly done by software developers.
No knowledge of implementation is needed.	Knowledge of implementation is required.
It can be referred to as outer or external software testing.	It is the inner or the internal software testing.
It is a functional test of the software.	It is a structural test of the software.
This testing can be initiated based on the requirement specifications document.	This type of testing of software is started after a detail design document.
No knowledge of programming is required.	It is mandatory to have knowledge of programming.
It is the behavior testing of the software.	It is the logic testing of the software.
It is applicable to the higher levels of testing of software.	It is generally applicable to the lower levels of software testing.
It is also called closed testing.	It is also called as clear box testing.
It is least time consuming.	It is most time consuming.
It is not suitable or preferred for algorithm testing.	It is suitable for algorithm testing.
Can be done by trial and error ways and methods.	Data domains along with inner or internal boundaries can be better tested.
Example: Search something on google by using keywords	**Example:** By input to check and verify loops
Black-box test design techniques- • Decision table testing • All-pairs testing • Equivalence partitioning • Error guessing	**White-box test design techniques-** • Control flow testing • Data flow testing • Branch testing
Types of Black Box Testing: • Functional Testing • Non-functional testing • Regression Testing	**Types of White Box Testing:** • Path Testing • Loop Testing • Condition testing
It is less exhaustive as compared to white box testing.	It is comparatively more exhaustive than black box testing.

8/31/2023

Software Testing and Introduction to quality

Introduction

What is a software Testing?

Software Testing is a process to investigation about the quality.

OR

Process of finding defects i.e. variance between Expected and actual result.

1) finding defects comparing Expected & Actual
2) Also we are checking quality.

Why testing is necessary

Software testing is necessary to make sure the product or application is defect free, as per the customer specification.

Software testing identifies fault whose removal increases the software quality and increase the software reliability.

Testing efforts is directly proportional to the complexity of the program.

Software should be
- Error free as per requirements.
- Efficient
- Secured

Why
1) defect free
2) identifies fault whose removal increases quality of s/w.
3) S/W should be
   - Error free
   - Efficient
   - secured.

8/31/2023

Nature of Error

1. Ambigious (unclear)
2. Incomplete
3. faulty

The nature of errors

Specifications are a common source of faults

A software system has an overall specification, derived from requirements analysis

The specification for a component can be:
- ambiguous (unclear)
- incomplete
- Faulty

Any such problems should, of course, be detected and remedied by verification of the specification prior to development of the component

Contd....

But this verification cannot and will not be totally effective

This is not all – there are other problems with specifications. Duri programming, the developer of a component may misunderstand the component specification.

The next type of error is where a component contain faults so tha it does not meet its specification. This may be due to two kinds of problem:
1. errors in the logic of the code – an error of commission
2. code that fails to meet all aspects of the specification – an e of omission

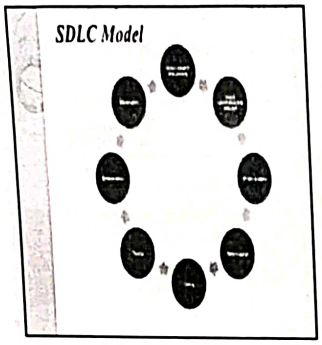
Finally, the kinds of errors that can arise in the coding of a component are:
- data not initialized
- loops repeated an incorrect number of times
- boundary value errors

- error in logic
- code fails to meet all aspects of specification
- data not initialized
- Repeated loop
- boundry value error

Software Development Life Cycle

- The software development lifecycle is used to facilitate the development of a large software product in a systematic, well-defined, and cost-effective way.
- An information system goes through a series of phases from conception to implementation. This process is called the Software Development Life-Cycle
- Various reasons for using a life-cycle model include:
  - Helps to understand the entire process
  - Enforces a structured approach to development
  - Enables planning of resources at each step
  - Enables subsequent controls of them
  - Aids management to track progress of the system
- The SDLC consists of several phases and these phases need to be identified along with defining the entry and exit criteria for every phase
- SDLC can be divided into 5-9 phases

8/31/2023



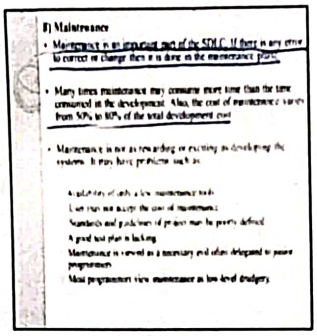
1) Project initiation and planning/ Recognition of need/ Preliminary investigation

- It is a brief investigation of the system under considerati
- This investigation provides the organization's computer steering committee and any project team a set of terms of references for more detailed work
- This is carried out by a senior manager and will result in study proposal.
- At this stage the need for changes in the existing system identified and shortcomings of the existing system are detected.

2) Project identification and selection Feasibility study

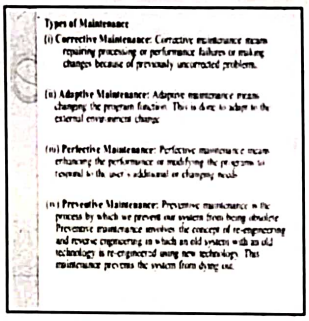
3

8/31/2023



f) Maintenance

- Maintenance is an important part of the SDLC. If there is any error to correct or change then it is done in the maintenance phase.
- Many times maintenance may consume more time than the time consumed in the development. Also, the cost of maintenance varies from 50% to 80% of the total development cost
- Maintenance is not as rewarding or exciting as developing the system. It may have problems such as:
  - A plethora of aids, like maintenance tools
  - User may not accept the cost of maintenance
  - Standards and guidelines of project may be poorly defined
  - A good test plan is lacking
  - Maintenance is viewed as a necessary evil often delegated to junior programmers
  - Most programmers view maintenance as low-level drudgery



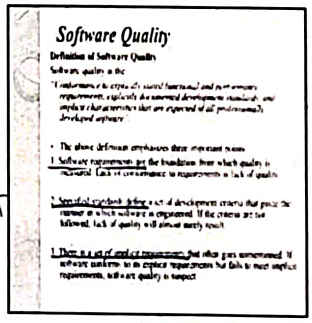
Types of Maintenance

(i) Corrective Maintenance: Corrective maintenance means repairing processing or performance failures or making changes because of previously uncorrected problems.

(ii) Adaptive Maintenance: Adaptive maintenance means changing the program function. This is done to adapt to the external environment change.

(iii) Perfective Maintenance: Perfective maintenance means enhancing the performance or modifying the programs to respond to the user's additional or changing needs

(iv) Preventive Maintenance: Preventive maintenance is the process by which we prevent our system from being obsolete. Preventive maintenance involves the concept of re-engineering and reverse engineering in which an old system with an old technology is re-engineered using new technology. This maintenance prevents the system from dying out.



Software Quality

Definition of Software Quality

Software quality is the

"Conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software."

- The above definition emphasizes three important points

1. Software requirements are the foundation from which quality is measured. Lack of conformance to requirements is lack of quality.

2. Specified standards define a set of development criteria that guide the manner in which software is engineered. If the criteria are not followed, lack of quality will almost surely result.

3. There is a set of implicit requirements that often goes unmentioned. If software conforms to its explicit requirements but fails to meet implicit requirements, software quality is suspect.

soft: requ
specified std
set of implicit
requirements

conformance to explicitely stated functional & performance requirements, explicitely document development stds, and implicit characteristics that are expected of all professionally developed s/w.

explicit Requirement: - Design specifications

implicit requirement → usability, performance, security, availability

8/31/2023

Classification of Software Qualities

- **External versus Internal Qualities**
  The external qualities are visible to the users of the system; the internal qualities are those that concern the developers of the system. For example, the internal quality of verifiability is necessary for achieving the external quality of reliability
- **Product and Process Qualities**
  We use a process to produce the software product. For example, if the process requires careful planning of system test data before any design and development of the system starts, products reliability will increase.

external Qualities
→ visible to users of system.

internal Qualities
→ concerning with developers

Product vs Process
→ process is used to produce product

Software Quality Attributes

Software quality is composed of six major attributes called characteristics:

Software Quality

Functionality | Reliability | Usability | Efficiency | Maintainability | Portability

1 Functionality: The capability to provide functions which meet stated and implied needs when the software is used
2 Reliability: The capability to maintain a specified level of performance
3 Usability: The capability to be understood, learned, and used
4 Efficiency: The capability to provide appropriate performance relative to the amount of resources used
5 Maintainability: The capability to be modified for purposes of making corrections, improvements, or adaptation
6 Portability: The capability to be adapted for different specified environments without applying actions or means other than those provided for this purpose in the product

S/W Quality Attributes
1. Functionality :- capability to provide functions
2. Reliability :- maintain level of performance
3. Usability :- understood, learn, use
4. Efficiency :- performance
5. Maintainability :- making corrects improvement
6. Portability :- capability of adaption diff. environment

Quality assurance

- Quality assurance means ensuring that a software system meets its quality goals
- To achieve its goals, a project must use effective tools and methods
- QA checks must be carried out during the development process at every available opportunity to see that the process is being carried out correctly
- To be effective, quality assurance must be planned in advance – along with the planning of all other aspects of a software project. The project manager

1 decides which quality factors are important for the particular project
2 selects standards and procedures from the quality manual that are appropriate to meeting the quality goals
3 assembles these into a quality assurance plan for the project. This describes what the procedures and standards are, when they will be done, and who does them.

Quality Assurance :-
→ means ensuring that the S/W system meets its quality Goals.
→ to achive its goal, project must use effective tools & methods
→ checklist must be there.

For Achiving Qualities
→ decides quality factor
→ select std & procedures.

8/31/2023

Quality Control

A quality control is an activity that checks that the project's quality factors are being achieved and produces some documentary evidence.

Quality Management

Quality management is the act of overseeing all activities and tasks that must be accomplished to maintain a desired level of excellence.

Quality control:—

→ procedure that focuses on fulfilling the quality requirements
→ identified & fix the defect
→ its a Reactive measure

Quality management:—

→ overseeing all activities & tasks that must be accomplished to maintain desired level of Excellence.

Difference between Quality Assurance (QA) and Quality Control (QC)

Quality Assurance (QA)	Quality Control (QC)
1 It is a procedure that focuses on providing assurance that quality requested will be achieved	It is a procedure that focuses on fulfilling the quality requirement
2 QA aims to prevent the defect	QC aims to identify and fix defects
3 It's a Preventive technique	It's a Corrective technique
4 It's a Proactive measure	It's a Reactive measure
It is the procedure to create the deliverables	It is the procedure to verify that deliverables
5 QA involves in full software development life cycle	QC involves in full software testing life cycle
6 In order to meet the customer requirements, QA defines standards and methodologies	QC confirms that the standards are followed while working on the project

The main motive is to prevent defects in the system. It is a very time-consuming activity	The main motive is to identify defects or bugs in the system. It is a more time-consuming activity
QA ensures that everything is executed in the right way and that is why it falls under verification activity	QC ensures that whatever we have done is as per the requirement, and that is why it falls under validation activity
5 It requires the involvement of the whole team	It requires the involvement of the testing team
The statistical technique applied for QA is known as SPC or Statistical Process Control (SPC)	The statistical technique applied in QC is known as SQC or Statistical Quality Control
6 It is performed before Quality Control	It is performed only after QA activity is done

8/31/2023

SOFTWARE QUALITY ASSURANC

- "Software Quality Assurance is a set o activities designed to evaluate the proc by which software is developed and/or maintained".
- "Quality assurance is a planned and systematic pattern of all actions neces to provide adequate confidence that th item or product conforms to establishe technical requirements (IEE83)".

SQA :-
S/w quality Assurance is a set of activities designed to evaluate the product by which S/w is developed and/or maintained

The process of the SQA:

1. Defines the requirements for software controlled system fault failure detection, isolation, and recovery;
2. Reviews the software-development processes and products for software error prevention and/or controlled change to reduced functionality states;
3. Defines the process for measuring and analyzing defects as well as reliability and maintainability factors.

SQA Objectives

The various objectives of SQA are as follows.

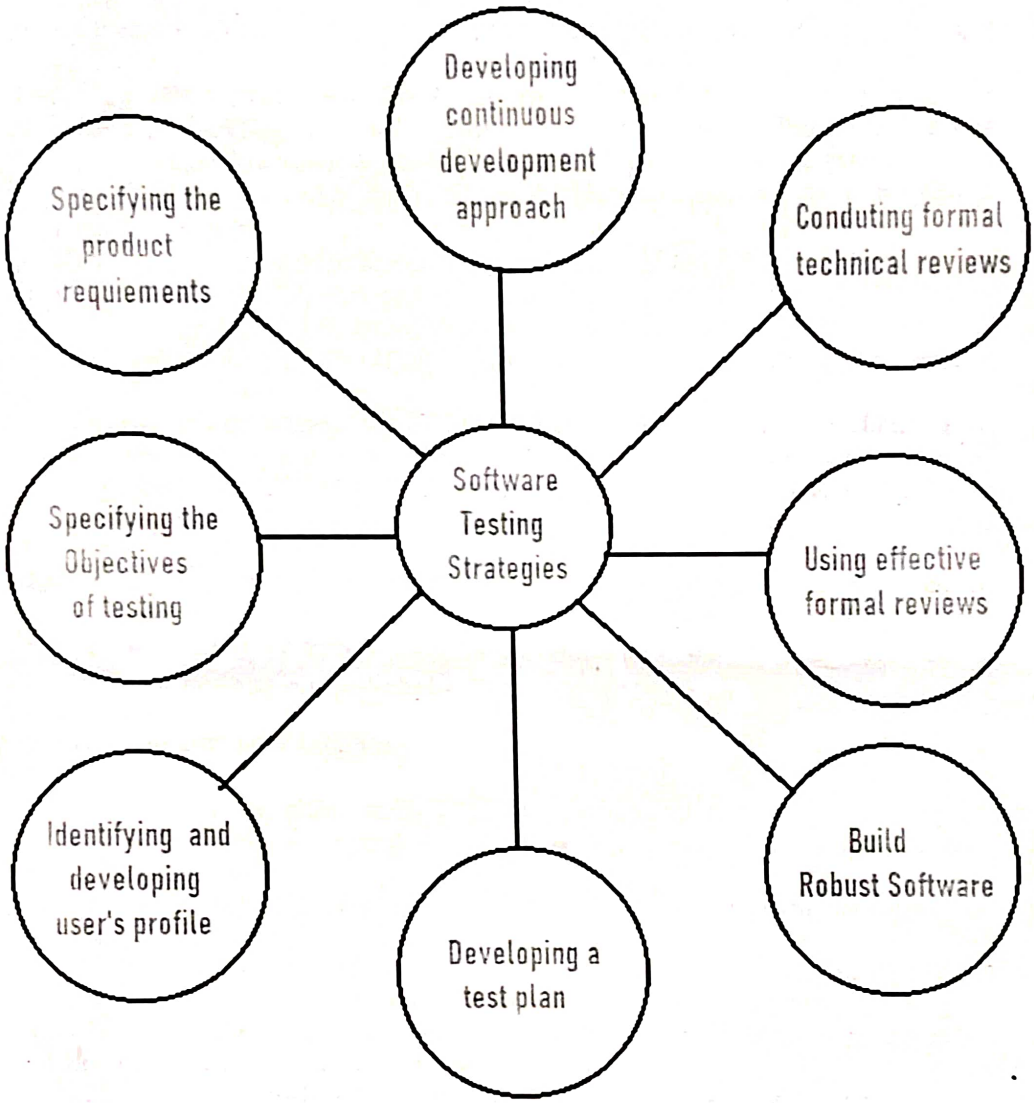
- Quality management approach
- Measurement and reporting mechanisms
- Effective software-engineering technology
- A procedure to assure compliance with software development standards where applicable
- A multi-testing strategy is drawn
- Formal technical reviews that are applied throughout the software process

9

Unit :-3

Strategic Approach to Software Testing

The main objective of software testing is to design the tests in such a way that it systematically finds different types of errors without taking much time and effort so that less time is required for the development of the software. The overall strategy for testing software includes:



1. **Before testing starts, it's necessary to identify and specify the requirements of the product in a quantifiable manner.** Different characteristics quality of the software is there such as maintainability that means the ability to update and modify, the probability that means to find and estimate any risk, and usability that means how it can easily be used by the customers or end-users. All these characteristic qualities should be specified in a particular order to obtain clear test results without any error.
2. **Specifying the objectives of testing in a clear and detailed manner.** Several objectives of testing are there such as effectiveness that means how effectively the software can achieve the target, any failure that means inability to fulfill the requirements and perform functions, and the cost of defects or errors that mean the

cost required to fix the error. All these objectives should be clearly mentioned in the test plan.

3. **For the software, identifying the user's category and developing a profile for each user.** Use cases describe the interactions and communication among different classes of users and the system to achieve the target. So as to identify the actual requirement of the users and then testing the actual use of the product.
4. **Developing a test plan to give value and focus on rapid-cycle testing.** Rapid Cycle Testing is a type of test that improves quality by identifying and measuring the any changes that need to be required for improving the process of software. Therefore, a test plan is an important and effective document that helps the tester to perform rapid cycle testing.
5. **Robust software is developed that is designed to test itself.** The software should be capable of detecting or identifying different classes of errors. Moreover, software design should allow automated and regression testing which tests the software to find out if there is any adverse or side effect on the features of software due to any change in code or program.
6. **Before testing, using effective formal reviews as a filter.** Formal technical reviews is technique to identify the errors that are not discovered yet. The effective technical reviews conducted before testing reduces a significant amount of testing efforts and time duration required for testing software so that the overall development time of software is reduced.
7. **Conduct formal technical reviews to evaluate the nature, quality or ability of the test strategy and test cases.** The formal technical review helps in detecting any unfilled gap in the testing approach. Hence, it is necessary to evaluate the ability and quality of the test strategy and test cases by technical reviewers to improve the quality of software.
8. **For the testing process, developing a approach for the continuous development.** As a part of a statistical process control approach, a test strategy that is already measured should be used for software testing to measure and control the quality during the development of software.

Advantages of software testing:

1. Improves software quality and reliability – Testing helps to identify and fix defects early in the development process, reducing the risk of failure or unexpected behavior in the final product.
2. Enhances user experience – Testing helps to identify usability issues and improve the overall user experience.
3. Increases confidence – By testing the software, developers and stakeholders can have confidence that the software meets the requirements and works as intended.
4. Facilitates maintenance – By identifying and fixing defects early, testing makes it easier to maintain and update the software.
5. Reduces costs – Finding and fixing defects early in the development process is less expensive than fixing them later in the life cycle.

Disadvantages of software testing:

1. Time-consuming – Testing can take a significant amount of time, particularly if thorough testing is performed.
2. Resource-intensive – Testing requires specialized skills and resources, which can be expensive.
3. Limited coverage – Testing can only reveal defects that are present in the test cases, and it is possible for defects to be missed.
4. Unpredictable results – The outcome of testing is not always predictable, and defects can be hard to replicate and fix.
5. Delays in delivery – Testing can delay the delivery of the software if testing takes longer than expected or if significant defects are identified.

Unit Testing | Software Testing

Unit testing is a type of software testing that focuses on individual units or components of a software system. The purpose of unit testing is to validate that each unit of the software works as intended and meets the requirements. Unit testing is typically performed by developers, and it is performed early in the development process before the code is integrated and tested as a whole system.

Unit tests are automated and are run each time the code is changed to ensure that new code does not break existing functionality. Unit tests are designed to validate the smallest possible unit of code, such as a function or a method, and test it in isolation from the rest of the system. This allows developers to quickly identify and fix any issues early in the development process, improving the overall quality of the software and reducing the time required for later testing.

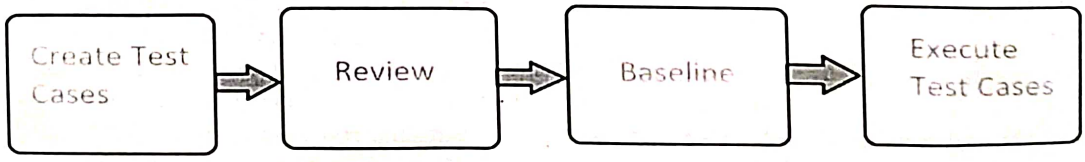
Unit Testing is a software testing technique by means of which individual units of software i.e. group of computer program modules, usage procedures, and operating procedures are tested to determine whether they are suitable for use or not. It is a testing method using which every independent module is tested to determine if there is an issue by the developer himself. It is correlated with the functional correctness of the independent modules. Unit Testing is defined as a type of software testing where individual components of a software are tested. Unit Testing of the software product is carried out during the development of an application. An individual component may be either an individual function or a procedure. Unit Testing is typically performed by the developer. In SDLC or V Model, Unit testing is the first level of testing done before integration testing. Unit testing is such a type of testing technique that is usually performed by developers. Although due to the reluctance of developers to test, quality assurance engineers also do unit testing.

Objective of Unit Testing:

The objective of Unit Testing is:

1. To isolate a section of code.
2. To verify the correctness of the code.
3. To test every function and procedure.
4. To fix bugs early in the development cycle and to save costs.
5. To help the developers to understand the code base and enable them to make changes quickly.
6. To help with code reuse.

Workflow of Unit Testing:



Unit Testing Tools:

Here are some commonly used Unit Testing tools:

1. Jtest
2. Junit

3. NUnit
4. EMMA
5. PHPUnit

Advantages of Unit Testing:

1. Unit Testing allows developers to learn what functionality is provided by a unit and how to use it to gain a basic understanding of the unit API.
2. Unit testing allows the programmer to refine code and make sure the module works properly.
3. Unit testing enables testing parts of the project without waiting for others to be completed.
4. Early Detection of Issues: Unit testing allows developers to detect and fix issues early in the development process, before they become larger and more difficult to fix.
5. Improved Code Quality: Unit testing helps to ensure that each unit of code works as intended and meets the requirements, improving the overall quality of the software.
6. Increased Confidence: Unit testing provides developers with confidence in their code, as they can validate that each unit of the software is functioning as expected.
7. Faster Development: Unit testing enables developers to work faster and more efficiently, as they can validate changes to the code without having to wait for the full system to be tested.
8. Better Documentation: Unit testing provides clear and concise documentation of the code and its behavior, making it easier for other developers to understand and maintain the software.
9. Facilitation of Refactoring: Unit testing enables developers to safely make changes to the code, as they can validate that their changes do not break existing functionality.
10. Reduced Time and Cost: Unit testing can reduce the time and cost required for later testing, as it helps to identify and fix issues early in the development process.

Disadvantages of Unit Testing:

1. The process is time-consuming for writing the unit test cases.
2. Unit Testing will not cover all the errors in the module because there is a chance of having errors in the modules while doing integration testing.
3. Unit Testing is not efficient for checking the errors in the UI(User Interface) part of the module.
4. It requires more time for maintenance when the source code is changed frequently.
5. It cannot cover the non-functional testing parameters such as scalability, the performance of the system, etc.
6. Time and Effort: Unit testing requires a significant investment of time and effort to create and maintain the test cases, especially for complex systems.
7. Dependence on Developers: The success of unit testing depends on the developers, who must write clear, concise, and comprehensive test cases to validate the code.
8. Difficulty in Testing Complex Units: Unit testing can be challenging when dealing with complex units, as it can be difficult to isolate and test individual units in isolation from the rest of the system.
9. Difficulty in Testing Interactions: Unit testing may not be sufficient for testing interactions between units, as it only focuses on individual units.
10. Difficulty in Testing User Interfaces: Unit testing may not be suitable for testing user interfaces, as it typically focuses on the functionality of individual units.
11. Over-reliance on Automation: Over-reliance on automated unit tests can lead to a false sense of security, as automated tests may not uncover all possible issues or bugs.
12. Maintenance Overhead: Unit testing requires ongoing maintenance and updates, as the code and test cases must be kept up-to-date with changes to the software.

What Is Integration Testing?

Integration testing is known as the second level of the software testing process, following unit testing. Integration testing involves checking individual components or units of a software project to expose defects and problems to verify that they work together as designed.

As a rule, the usual software project consists of numerous software modules, many of them built by different programmers. Integration testing shows the team how well these disparate elements work together. After all, each unit may function perfectly on its own, but the pressing question is, "But can they be brought together and work smoothly?"

So, integration testing is the way we find out if the various parts of a software application can play well with others!

Become a software testing expert with Simplilearn's specialized software testing courses. Gain practical knowledge, master testing techniques, and excel in the dynamic IT industry.

Why Perform Integration Testing?

Other than the fundamental truth that developers must test all software applications before releasing them to the public, there are some specific reasons why developers should perform integration testing.

- Incompatibility between software modules can cause errors
- Developers must confirm that every software module can interact with the database
- Requirements change, thanks to client input. However, maybe those new requirements haven't been thoroughly tested yet and should be
- Every software developer has their understanding and programming logic. Integration testing ensures that these various units function smoothly
- There may be potential problems with hardware compatibility
- Modules often interact with third-party APIs or tools, so we need integration testing to verify that the data these tools accept is correct

Advantages of Integration Testing

- Integration testing ensures that every integrated module functions correctly
- Integration testing uncovers interface errors
- Testers can initiate integration testing once a module is completed and doesn't require waiting for another module to be done and ready for testing
- Testers can detect bugs, defects, and security issues
- Integration testing provides testers with a comprehensive analysis of the whole system, dramatically reducing the likelihood of severe connectivity issues

Challenges of Integration Testing

Unfortunately, integration testing has some difficulties to overcome as well.

- If testing involves dealing with two different systems created by two different vendors, there will be questions about how these components will affect and interact with each other

- Integrating new and legacy systems demands many testing efforts and potential changes
- Integration testing becomes complex due to the variety of components involved (e.g., platforms, environments, databases)
- Integration testing requires testing not only the integration links but the environment itself, adding another layer of complexity to the process

Guidelines for Integration Testing

If you are attempting integration testing, keep these guidelines in mind:

- First, don't initiate integration testing until each module has undergone functional testing first
- Module testing should follow an accepted sequence not to overlook any integration scenarios
- Agree on a test case strategy to prepare and execute test cases in conjunction with the test data
- Study the application's architecture and structure, identifying the most important modules to be tested first. Also, identify every possible scenario
- Design test cases that will create detailed interface verification
- Input data is vital to conducting reliable integration testing, so choose test case execution input data wisely
- Generate reports on any bugs found. Send bug reports to the developers, have them fix the errors, then conduct testing again

What Is Validation Testing In Software Development?

Validation testing is the process of assessing a new software product to ensure that its performance matches consumer needs. Product development teams might perform validation testing to learn about the integrity of the product itself and its performance in different environments.

Developers can perform validation testing themselves, or collaborate with quality assurance professionals, external validation testing professionals or clients to identify elements of the code to improve. Developers can also combine this type of testing with other useful techniques like product verification, debugging and certification to help ensure the product is ready for the market.

Why is validation testing important?

Software validation can help product development teams ensure their product can satisfy customer needs and expectations. Validation testing can also help software developers identify and fix coding bugs or address other areas of improvement before launching the product.

Phases of validation testing

Validation testing is a complex process that involves finding and testing every user need or requirement to ensure they function well. Here are the basic phases of validation testing:

1. Design qualification

The process of design qualification, or DQ, includes creating a list of end-user business requirements and designing a validation testing plan to address them before launching the product. This plan can also be a useful written record of the design specifications that the developer and consumer desire. After writing the testing plan, development teams can seek approval from managers or shareholders before they begin the testing process.

2. Installation qualification

Installation qualification, or IQ, involves installing the software according to the validation testing plan. Product development teams may ensure that both system hardware and the installation process itself match the design specifications. This phase also involves ensuring that the test environment is suitable for product operation and matches the environment in which the product is likely to perform once the company releases it to the public.

3. Operational qualification

Operational qualification, or OQ, involves testing the product with a variety of testing operations to ensure the product meets the specified user requirements. Important validation testing techniques include unit testing, integration testing and system testing. These are all different types of functionality testing, which can determine if various elements of the software function according to the user requirements.

Software development teams can follow the plan from the design qualification phase to ensure they test every product specification in the appropriate environment. It's useful for product development team members to record these tests carefully to create a written record

of the software performance. They can also record the creation of deliverables and ensure those deliverables are satisfactory through a deliverable approval process.

4. Performance qualification

Performance qualification, or PQ, testing verifies that a product can perform according to business needs in the real world. Developers on the internal team can perform alpha testing to assess the functionality of the software under simulated real-world conditions. After performing their own testing, a product development team can offer clients the chance to test the product through a process called beta testing.

In beta testing, an external client can use the product and identify bugs or technical challenges. They can then report this information back to the development team, allowing them the opportunity to make changes before releasing the product to the public. After receiving approval from the beta testers, the product may be ready for production and deployment.

5. Production

After completing all levels of validation testing, a software product may go into production. This means that the product is ready to be marketed and sold to consumers. The software development team might help facilitate the deployment and installation process. The company can also offer technical assistance to individuals experiencing minor technical challenges in the product. If they discover a major technical bug, the development team can address it by devising a solution and release a software update.

Types of validation testing

Here are four important types of validation testing and how they work:

Unit testing

Unit testing is a form of validation testing that involves assessing small pieces of code individually. Units can include pieces of code like functions, methods, procedures, modules or objects. Testing these units separately from each other can help ensure that each is performing well. This improves the chance of the software functioning well overall. Developers can also use the unit testing process to create records of the units and their functions, which may be helpful for future software updates.

There are different techniques that can help developers perform unit testing. One unit testing technique is known as black box testing. This involves testing a unit without prior knowledge of its code or functionality. This is the inverse of white box testing, also known as glass box testing or clear box testing.

In white box testing, developers use their knowledge of internal data structures and source code software architecture to test unit functionality. Both black box and white box testing are techniques that developers may use for both unit testing and other validation testing procedures.

Integration testing

After performing unit testing to if each unit of code within the software is functioning correctly, software development teams can run integration testing to learn about how well the units function once they integrate them together into a larger system. Specifically, developers can ensure that data flow across modules is successful.

The two main types of integration testing are the top-down approach and the bottom-up approach. Both techniques involve testing the integration of software modules in a systematic, step-by-step process.

In the top-down approach, developers start by connecting high-level modules and work their way through the control flow of the software architectures structure from top to bottom. Bottom-up software testing involves testing low-level elements first and working up to the top of the architecture structure.

System testing

Also known as system-level testing or system-integration testing, this type of validation testing can assess the software as a complete system. This can help confirm that the product functions according to the end-to-end system specifications. Most forms of system testing involve black box validation testing techniques. Here are three specific subcategories of system testing technique:

- Smoke testing: Smoke testing involves testing the essential functional elements of a software product. For example, smoke testing for a music library application might involve assessing the user's ability to log in, search music, listen to music and add it to a personal music library.
- Sanity testing: After smoke testing, developers may make changes to the software code, then perform sanity testing to assess the functionality of the software following those code changes.
- Regression testing: Regression testing is the final step in the system testing process. It involves finding and fix any new bugs that may arise from changes made in the previous testing steps to ensure that the entire software system still functions as it should.

User acceptance testing

User acceptance testing, or UAT, is a form of performance qualification beta testing that invites a client to test the product and ensure that it meets their needs. This is often the last stage in the validation testing process and can be helpful because it tests the product in its proper environment. It can also help reveal challenges that the developers may not have noticed because they're already so familiar with the product.

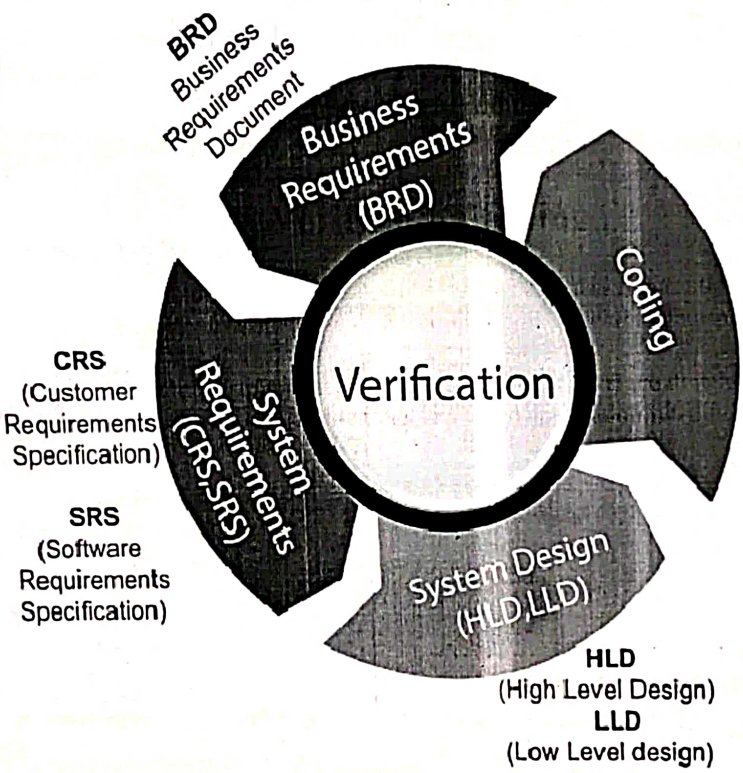
Verification and Validation Testing

In this section, we will learn about verification and validation testing and their major differences.

Verification testing

Verification testing includes different activities such as business requirements, system requirements, design review, and code walkthrough while developing a product.

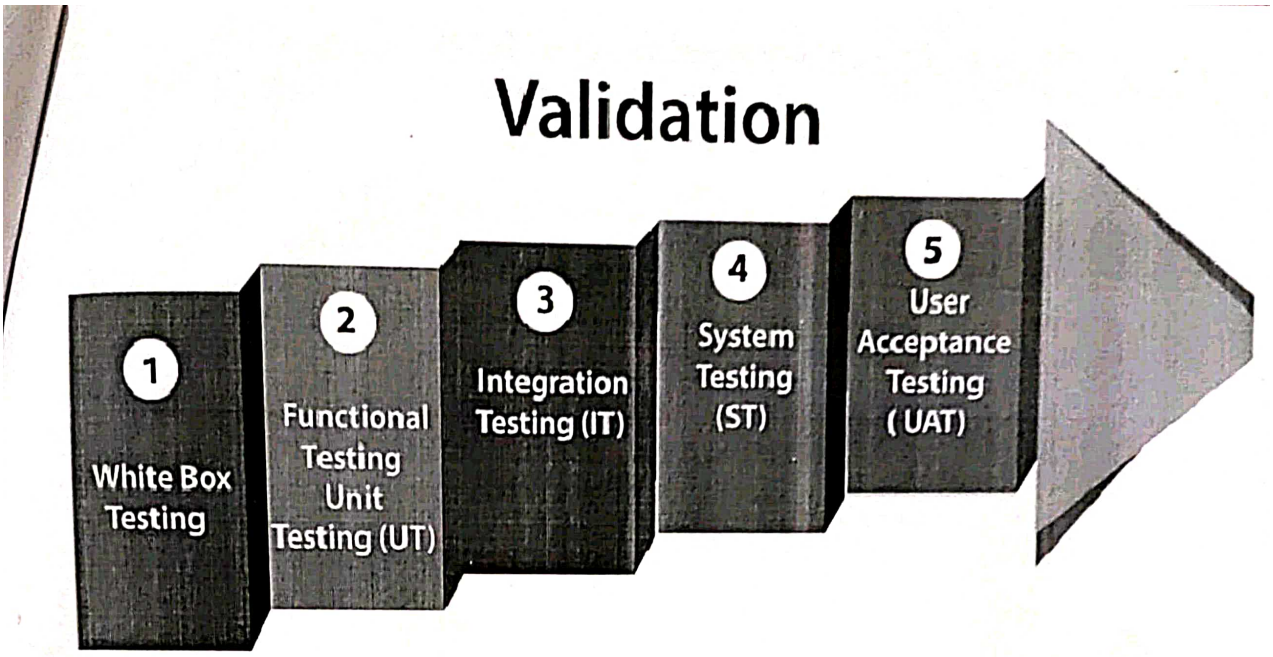
It is also known as static testing, where we are ensuring that "**we are developing the right product or not**". And it also checks that the developed application fulfilling all the requirements given by the client.



Validation testing

Validation testing is testing where tester performed functional and non-functional testing. Here **functional testing** includes Unit Testing (UT), Integration Testing (IT) and System Testing (ST), and **non-functional** testing includes User acceptance testing (UAT).

Validation testing is also known as dynamic testing, where we are ensuring that "**we have developed the product right.**" And it also checks that the software meets the business needs of the client.



Note: Verification and Validation process are done under the V model of the software development life cycle.

Difference between verification and validation testing

Verification	Validation
We check whether we are developing the right product or not.	We check whether the developed product is right.
Verification is also known as **static testing**.	Validation is also known as **dynamic testing**.
Verification includes different methods like Inspections, Reviews, and Walkthroughs.	Validation includes testing like functional testing, system testing, integration, and User acceptance testing.
It is a process of checking the work-products (not the final product) of a development cycle to decide whether the product meets the specified requirements.	It is a process of checking the software during or at the end of the development cycle to decide whether the software follow the specified business requirements.
Quality assurance comes under verification testing.	**Quality control** comes under validation testing.
The execution of code does not happen in the verification testing.	In validation testing, the execution of code happens.
In verification testing, we can find the bugs early in the development phase of the product.	In the validation testing, we can find those bugs, which are not caught in the verification process.
Verification testing is executed by the Quality assurance team to make sure that the product is developed according to customers' requirements.	Validation testing is executed by the testing team to test the application.
Verification is done before the validation testing.	After verification testing, validation testing takes place.
In this type of testing, we can verify that the inputs follow the outputs or not.	In this type of testing, we can validate that the user accepts the product or not.

Defect Management Process

The Defect Management Process is a systematic approach to identify, track, and resolve defects in software development. It typically includes the following steps:

1. **Defect identification** – Defects are identified through various testing activities, such as unit testing, integration testing, and user acceptance testing.
2. **Defect logging** – Defects are logged in a defect tracking system, along with details such as description, severity, and priority.
3. **Defect triage** – The triage process involves evaluating the defects to determine their priority and the resources required to resolve them.
4. **Defect assignment** – Defects are assigned to developers or testers for resolution, based on their expertise and availability.
5. **Defect resolution –** The assigned personnel work on resolving the defects by fixing the code, updating the documentation, or performing other necessary actions.
6. **Defect verification** – Once the defect is resolved, it is verified by the tester to ensure that it has been fixed correctly and does not introduce any new defects.
7. **Defect closure** – Once the defect has been verified, it is closed and the status is updated in the defect tracking system.
8. **Defect reporting** – Regular reports on the status of defects, including the number of open defects, the number of defects resolved, and the average time to resolve defects, are generated to provide visibility into the defect management process.

Defect Report:

A defect report is a document that has concise details about what defects are identified, what action steps make the defects show up, and what are the expected results instead of the application showing error (defect) while taking particular step by step actions.
Defect reports are usually created by the Quality Assurance team and also by the end-users (customers). Often customers detect more defects and report them to the support team of the software development since the majority of the customers curiously tries out every feature in the application. Now, you know what actually defect and defect reports are.
It is usual for QA teams to get defect reports from the clients that are either too short to reproduce and rectify or too long to understand what actually went wrong.
For example,
Defect Description: The application doesn't work as expected.
Now, how in the world does a developer or QA know what went wrong which doesn't meet the client expectation?
In such a case, the developer report to the QA that he couldn't find any problem or he may have fixed any other error but not the actual one client detected. So that's why it's really important to create a concise defect report to get bugs fixed.
All right. You have a pretty good idea about what, whys and how's of a defect report. So it's time for what is inside the report.
A typical defect report contains the information in an xls Sheet as follows.

1. Defect ID :

Nothing but a serial number of defects in the report.

2. Defect Description :

A short and clear description of the defect detected.

3. Action Steps :

What the client or QA did in an application that results in the defect. Step by step actions they took.

4. Expected Result :

What results are expected as per the requirements when performing the action steps mentioned.

5. Actual Result :

What results are actually showing up when performing the action steps.

6. Severity :

Trivial (A small bug that doesn't affect the software product usage).

1. Low –
   A small bug that needs to be fixed and again it's not going to affect the performance of the software.
2. Medium –
   This bug does affect the performance. Such as being an obstacle to do a certain action. Yet there is another way to do the same thing.
3. High –
   It highly impacts the software though there is a way around to successfully do what the bug cease to do.
4. Critical –
   These bugs heavily impacts the performance of the application. Like crashing the system, freezes the system or requires the system to restart for working properly.

7. Attachments :

A sequence of screenshots of performing the step by step actions and getting the unexpected result. One can also attach a short screen recording of performing the steps and encountering defects. Short videos help developers and/or QA to understand the bugs easily and quickly.

8. Additional information :

The platform you used, operating system and version. And other information which describes the defects in detail for assisting the developer understand the problem and fixing the code for getting desired results.

Software Engineering | Software Quality Assurance

Unit :-5

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 (SQA) encompasses

- SQA process
- specific quality assurance and quality control tasks (including technical reviews and a multitiered testing strategy)
- effective software engineering practice (methods and tools)
- control of all software work products and the changes made to them
- a procedure to ensure compliance with software development standards (when applicable)
- measurement and reporting mechanisms

Elements Of Software Quality Assurance:

1. **Standards:** The IEEE, ISO, and other standards organizations have produced a broad array of software engineering standards and related documents. The job of SQA is to ensure that standards that have been adopted are followed an all work products conform to them.

2. **Reviews and audits:** Technical reviews are a quality control activity performed by software engineers for software engineers. Their intent is to uncover errors. Audits are a type of review performed by SQA personnel(people employed in an organization) with the intent of ensuring that quality guidelines are being followed for software engineering work.

3. **Testing:** Software testing is a quality control function that has one primary goal—to find errors. The job of SQA is to ensure that testing is properly planned and efficiently conducted for primary goal of software.

4. **Error/defect collection and analysis**: SQA collects and analyzes error and defect data to better understand how errors are introduced and what software engineering activities are best suited to eliminating them.

5. **Change management:** SQA ensures that adequate change management practices have been instituted.

6. **Education:** Every software organization wants to improve its software engineering practices. A key contributor to improvement is education of software engineers, their managers, and other stakeholders. The SQA organization takes the lead in software process improvement which is key proponent and sponsor of educational programs.

7. **Security management:** SQA ensures that appropriate process and technology are used to achieve software security.

8. **Safety:** SQA may be responsible for assessing the impact of software failure and for initiating those steps required to reduce risk.

9. **Risk management**: The SQA organization ensures that risk management activities are properly conducted and that risk-related contingency plans have been established.

Software quality assurance focuses on:

- software's portability
- software's usability
- software's reusability
- software's correctness
- software's maintainability
- software's error control

Software Quality Assurance has:

1. A quality management approach
2. Formal technical reviews
3. Multi testing strategy
4. Effective software engineering technology
5. Measurement and reporting mechanism

Major Software Quality Assurance Activities:

1. **SQA Management Plan:**
   Make a plan for how you will carry out the SQA throughout the project. Think about which set of software engineering activities are the best for project. check level of SQA team skills.

2. **Set The Check Points:**
   SQA team should set checkpoints. Evaluate the performance of the project on the basis of collected data on different check points.

3. **Multi testing Strategy:**
   Do not depend on a single testing approach. When you have a lot of testing approaches available use them.

4. **Measure Change Impact:**
The changes for making the correction of an error sometimes re introduces more errors keep the measure of impact of change on project. Reset the new change to change check the compatibility of this fix with whole project.

5. **Manage Good Relations:**
In the working environment managing good relations with other teams involved in the project development is mandatory. Bad relation of SQA team with programmers team will impact directly and badly on project. Don't play politics.

Benefits of Software Quality Assurance (SQA):

1. SQA produces high quality software.
2. High quality application saves time and cost.
3. SQA is beneficial for better reliability.
4. SQA is beneficial in the condition of no maintenance for a long time.
5. High quality commercial software increase market share of company.
6. Improving the process of creating software.
7. Improves the quality of the software.
8. It cuts maintenance costs. Get the release right the first time, and your company can forget about it and move on to the next big thing. Release a product with chronic issues, and your business bogs down in a costly, time-consuming, never-ending cycle of repairs.

Disadvantage of SQA:
There are a number of disadvantages of quality assurance. Some of them include adding more resources, employing more workers to help maintain quality and so much more.

Software Reviews

Software reviews are an essential part of the Software Development Life Cycle (SDLC).

A **software review** is a task in which a group of people tries to resolve the errors and defects in particular software.

Reviews play an important role in developing any software model.

Objectives of a software review

- We can use software reviews to discover the expectations that clients have from us.
- We can use them to identify and eliminate any errors and defects in the software.
- We can utilize them to reduce the gap between the clients' demands and the developer's supply of products.

Process of software review

Types of software reviews

There are three types of software reviews:

- *Software peer reviews*
- *Software management reviews*
- *Software audit reviews*

Software peer review

Generally, a **software peer review** is performed to detect any errors in the software and correct them. The quality of the software is also checked during this process.

A peer review may be categorized as:

- **Code review**: In this type of software peer review, the written code is scrutinized to detect any possible errors.
- **Pair programming**: In this type of peer review, two developers work on the same code and develop it.
- **Walkthrough**: In this type of peer review, the senior-most members guide the development team to work and correct errors and/or defects in the software.
- **Technical review**: In this type of peer review, the senior-most members of the team test the products and give their feedback on them.
- **Inspection**: In this type of peer review, the inspection process is carried out step by step to find any possible errors in the software.

Software management review

The **software management review** process involves a team of reviewers checking the status of the work that is performed by developers.

Advantages of a software management review

- A software management review helps us check the status of the work.
- We can use this type of review to see whether the project is up-to-date or not.

Software audit review

Software audit reviews are reviewed in which people from the development team participate in discussions and suggest ideas for the project. They also try to create solutions for any errors that they encounter in the software under discussion.

Advantages of a software audit review

- We can use software reviews to detect errors at an early stage of the software's development.
- Software reviews help us save time. in cases where an error is detected at an early stage of the software's development process.
- We can use software reviews to discover what the client's expectations are from the developers.

Formal Technical Reviews

- Formal Technical review is a software quality assurance activity performed by software engineer.

Objectives of FTR

1. FTR is useful to uncover error in logic, function and implementation for any representation of the software.
2. The purpose of FTR is to ensure that software meets specified requirements.
3. It is also ensure that software is represented according to predefined standards.
4. It helps to review the uniformity in software development process.
5. It makes the project more manageable.

- Besides the above mentioned objectives, the purpose of FTR is to enable junior engineer to observer the analysis, design, coding and testing approach more closely.
- Each FTR is conducted as meeting and is considered successfully only if it is properly planned, controlled and attended.

Steps in FTR

1. The review meeting

- Every review meeting should be conducted by considering the following constraints-

1. Involvement of people

   Between 3 and 5 people should be involve in the review.

2. Advance preparation Advance preparation should occur but it should be very short that is at the most 2 hours of work for each person can be spent in this preparation
3. Short duration The short duration of the review meeting should be less than two hour.

- Rather than attempting to review the entire design walkthrough are conducted for modules or for small group of modules.
- The focus of the FTR is on work product (a software component to be reviewed). The review meeting is attended by the review leader, all reviewers and the producer.
- The review leader is responsible for evaluating for product for its deadlines. The copies of product material is then distributed to reviewers. -The producer organises "walkthrough" the product, explaining the material, while the reviewers raise the issues based on theirs advance preparation.
- One of the reviewers become recorder who records all the important issues raised during the review. When error are discovered, the recorder notes each.
- At the end of the review, the attendees decide whether to accept the product or not, with or without modification.

2. Review reporting and record keeping

- During the FTR, the reviewer actively record all the issues that have been raised.
- At the end of meeting these all raised issues are consolidated and review issue list is prepared.
- Finally, formal technical review summary report is produced.

3. Review guidelines

- Guidelines for the conducting of formal technical review must be established in advance. These guidelines must be distributed to all reviewers. agreed upon. and then followed.
- For example.

Guideline for review may include following things

1. Concentrate on work product only. That means review the product not the producers.
2. Set an agenda of a review and maintain it.
3. When certain issues are raised then debate or arguments should be limited. Reviews should not ultimately results in some hard feelings.
4. Find out problem areas. but don't attempt to solve every problem noted.
5. Take written notes (it is for record purpose)
6. Limit the number of participants and insists upon advance preparation.
7. Develop a checklist for each product that is likely to be reviewed.
8. Allocate resources and time schedule for FTRs in order to maintain time schedule.
9. Conduct meaningful trainings for all reviewers in order to make reviews effective.
10. Reviews earlier reviews which serve as the base for the current review being conducted.

Difference between Inspection and Walkthrough

1.Walkthrough :

Walkthrough is a method of conducting informal group/individual review. In a walkthrough, author describes and explain work product in a informal meeting to his peers or supervisor to get feedback. Here, validity of the proposed solution for work product is checked.

It is cheaper to make changes when design is on the paper rather than at time of conversion. Walkthrough is a static method of quality assurance. Walkthrough are informal meetings but with purpose.

2.Inspection :

An inspection is defined as formal, rigorous, in depth group review designed to identify problems as close to their point of origin as possible. Inspections improve reliability, availability, and maintainability of software product.

Anything readable that is produced during the software development can be inspected. Inspections can be combined with structured, systematic testing to provide a powerful tool for creating defect-free programs.

Inspection activity follows a specified process and participants play well-defined roles.
An inspection team consists of three to eight members who plays roles of moderator, author, reader, recorder and inspector.

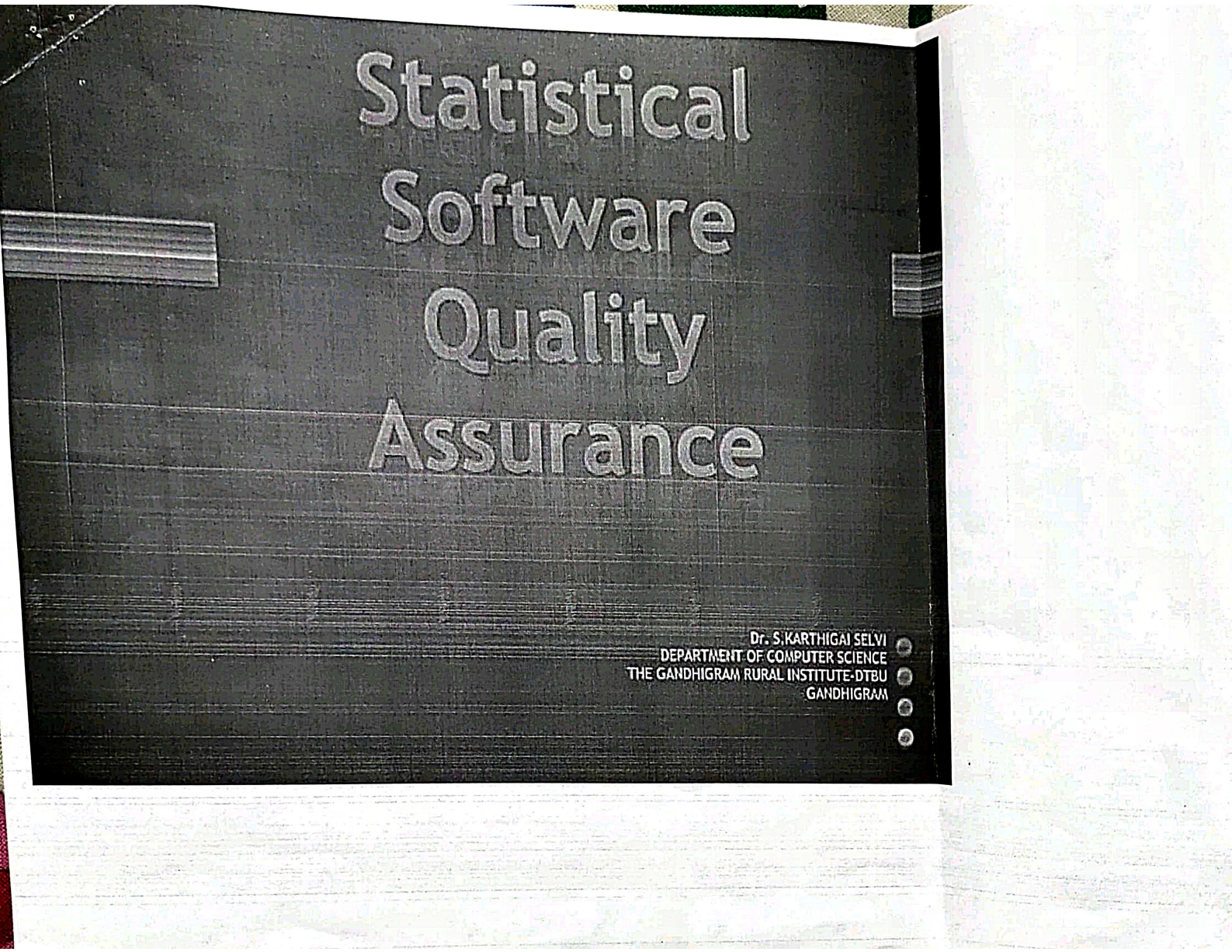
For example, designer can acts as inspector during code inspections while a quality assurance representative can act as standard enforcer.

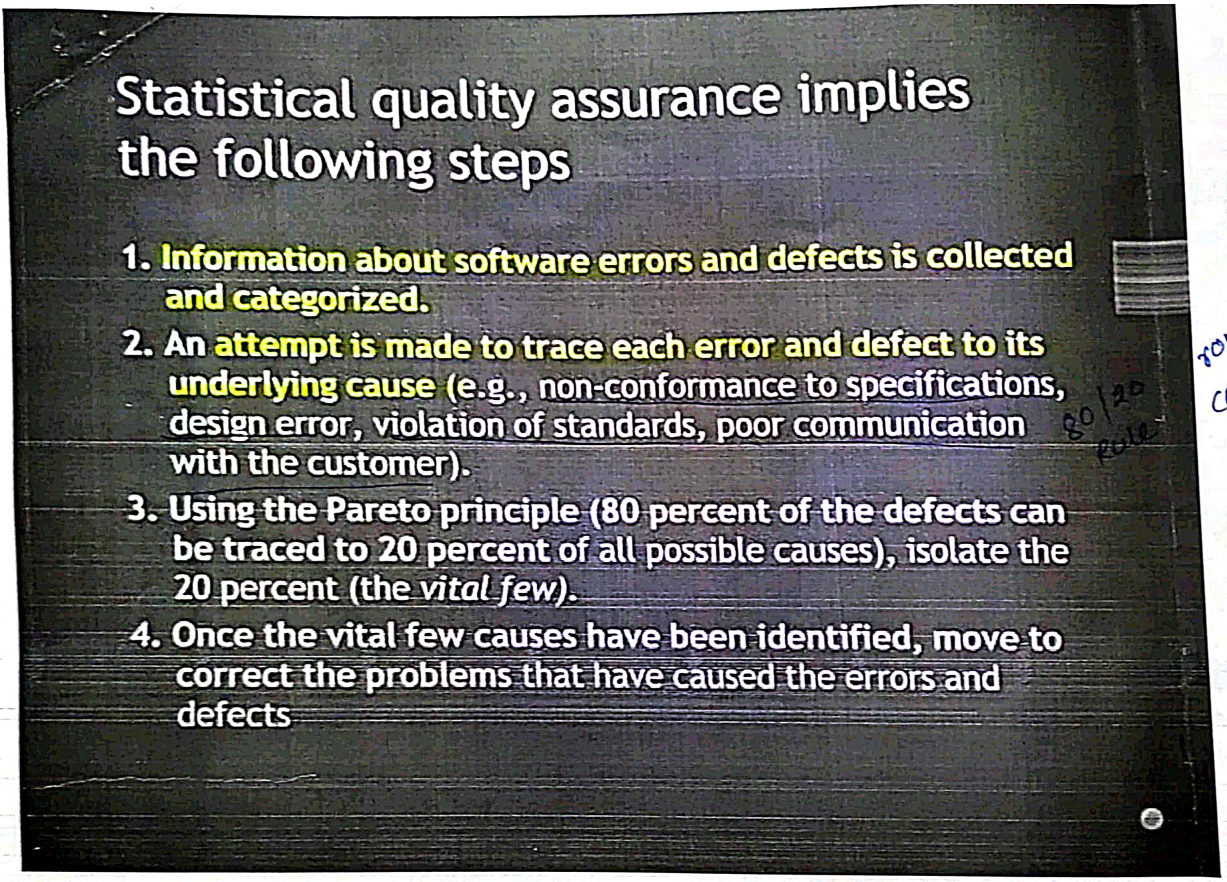
Stages in the inspections process :

- **Planning :** Inspection is planned by moderator.
- **Overview meeting :** Author describes background of work product.
- **Preparation :** Each inspector examines work product to identify possible defects.
- **Inspection meeting :** During this meeting, reader reads through work product, part by part and inspectors points out the defects for every part.
- **Rework :** Author makes changes to work product according to action plans from the inspection meeting.
- **Follow-up :** Changes made by author are checked to make sure that everything is correct.

Difference between Inspection and Walkthrough :

S.No.	Inspection	Walkthrough
1.	It is formal.	It is informal.
2.	Initiated by project team.	Initiated by author.
3.	A group of relevant persons from different departments participate in the inspection.	Usually team members of the same project take participation in the walkthrough. Author himself acts walkthrough leader.
4.	Checklist is used to find faults.	No checklist is used in the walkthrough.
5.	Inspection processes includes overview, preparation, inspection, and rework and follow up.	Walkthrough process includes overview, little or no preparation, little or no preparation examination (actual walkthrough meeting), and rework and follow up.
6.	Formalized procedure in each step.	No formalized procedure in the steps.
7.	Inspection takes longer time as list of items in checklist is tracked to completion.	Shorter time is spent on walkthrough as there is no formal checklist used to evaluate program.
8.	Planned meeting with the fixed roles assigned to all the members involved.	Unplanned
9.	Reader reads product code. Everyone inspects it and comes up with detects.	Author reads product code and his teammate comes up with the defects or suggestions.
10.	Recorder records the defects.	Author make a note of defects and suggestions offered by teammate.
11.	Moderator has a role that moderator making sure that the discussions proceed on the productive lines.	Informal, so there is no moderator.


Statistical
Software
Quality
Assurance
Dr. S.KARTHIGAI SELVI
DEPARTMENT OF COMPUTER SCIENCE
THE GANDHIGRAM RURAL INSTITUTE-DTBU
GANDHIGRAM


Statistical quality assurance implies the following steps
1. Information about software errors and defects is collected and categorized.
2. An attempt is made to trace each error and defect to its underlying cause (e.g., non-conformance to specifications, design error, violation of standards, poor communication with the customer).
3. Using the Pareto principle (80 percent of the defects can be traced to 20 percent of all possible causes), isolate the 20 percent (the vital few).
4. Once the vital few causes have been identified, move to correct the problems that have caused the errors and defects
80/20 rule
roughly 80% of effect comes from 20% of causes.

A software may encountered more defects and uncovered errors

- Incomplete or erroneous specifications (IES)
- Misinterpretation of customer communication (MCC)
- Intentional deviation from specifications (IDS)
- Violation of programming standards (VPS)
- Error in data representation (EDR)
- Inconsistent component interface (ICI)
- Error in design logic (EDL)
- Incomplete or erroneous testing (IET)
- Inaccurate or incomplete documentation (IID)
- Error in programming language translation of design (PLT)
- Ambiguous or inconsistent human/computer interface (HCI)
- Miscellaneous (MIS)

Statistical quality assurance techniques for software have been shown to provide substantial quality improvement [Art97]. In some cases, software organizations have achieved a 50 percent reduction per year in defects after applying these techniques

	Total		Serious		Moderate		Minor	
Error	No.	%	No.	%	No.	%	No.	%
IES	205	22%	34	27%	68	18%	103	24%
MCC	156	17%	12	9%	68	18%	76	17%
IDS	48	5%	1	1%	24	6%	23	5%
VPS	25	3%	0	0%	15	4%	10	2%
EDR	130	14%	26	20%	68	18%	36	8%
ICI	58	6%	9	7%	18	5%	31	7%
EDI	45	5%	14	11%	12	3%	19	4%
IET	95	10%	12	9%	35	9%	48	11%
IID	36	4%	2	2%	20	5%	14	3%
PIT	60	6%	15	12%	19	5%	26	6%
HCI	28	3%	3	2%	17	4%	8	2%
MIS	56	6%	0	0%	15	4%	41	9%
Totals	942	100%	128	100%	379	100%	435	100%

ISO 9000 Certification in Software Engineering

The International organization for Standardization is a world wide federation of national standard bodies. The **International standards organization (ISO)** is a standard which serves as a for contract between independent parties. It specifies guidelines for development of **quality system**.

Quality system of an organization means the various activities related to its products or services. Standard of ISO addresses to both aspects i.e. operational and organizational aspects which includes responsibilities, reporting etc. An ISO 9000 standard contains set of guidelines of production process without considering product itself.

ISO 9000 Certification

Why ISO Certification required by Software Industry?
There are several reasons why software industry must get an ISO certification. Some of reasons are as follows :

- This certification has become a standards for international bidding.
- It helps in designing high-quality repeatable software products.
- It emphasis need for proper documentation.
- It facilitates development of optimal processes and totally quality measurements.

Features of ISO 9001 Requirements :

- **Document control –**
  All documents concerned with the development of a software product should be properly managed and controlled.
- **Planning –**
  Proper plans should be prepared and monitored.
- **Review –**
  For effectiveness and correctness all important documents across all phases should be independently checked and reviewed .
- **Testing –**
  The product should be tested against specification.
- **Organizational Aspects –**
  Various organizational aspects should be addressed e.g., management reporting of the quality team.

Advantages of ISO 9000 Certification :
Some of the advantages of the ISO 9000 certification process are following :

- Business ISO-9000 certification forces a corporation to specialize in "how they are doing business". Each procedure and work instruction must be documented and thus becomes a springboard for continuous improvement.
- Employees morale is increased as they're asked to require control of their processes and document their work processes
- Better products and services result from continuous improvement process.
- Increased employee participation, involvement, awareness and systematic employee training are reduced problems.

Shortcomings of ISO 9000 Certification :
Some of the shortcoming of the ISO 9000 certification process are following :

- ISO 9000 does not give any guideline for defining an appropriate process and does not give guarantee for high quality process.
- ISO 9000 certification process have no international accreditation agency exists.

Six Sigma

Six Sigma is a set of methodologies and tools used to **improve business processes by reducing defects and errors, minimizing variation, and increasing quality and efficiency**. The goal of Six Sigma is to achieve a level of quality that is nearly perfect, with only 3.4 defects per million opportunities.

Six Sigma is a set of methodologies and tools used to **improve business processes by reducing defects and errors, minimizing variation, and increasing quality and efficiency**. The goal of Six Sigma is to achieve a level of quality that is nearly perfect, with only 3.4 defects per million opportunities.

Characteristics of Six Sigma:
The Characteristics of Six Sigma are as follows:

1. **Statistical Quality Control:**
   Six Sigma is derived from the Greek Letter ? which denote Standard Deviation in statistics. Standard Deviation is used for measuring the quality of output.
2. **Methodical Approach:**
   The Six Sigma is a systematic approach of application in DMAIC and DMADV which can be used to improve the quality of production. DMAIC means for Design-Measure- Analyze-Improve-Control. While DMADV stands for Design-Measure-Analyze-Design-Verify.
3. **Fact and Data-Based Approach:**
   The statistical and methodical method shows the scientific basis of the technique.
4. **Project and Objective-Based Focus:**
   The Six Sigma process is implemented to focus on the requirements and conditions.
5. **Customer Focus:**
   The customer focus is fundamental to the Six Sigma approach. The quality improvement and control standards are based on specific customer requirements.
6. **Teamwork Approach to Quality Management:**
   The Six Sigma process requires organizations to get organized for improving quality.

Six Sigma Methodologies:
Two methodologies used in the Six Sigma projects are DMAIC and DMADV.

- **DMAIC** is used to enhance an existing business process. The DMAIC project methodology has five phases:
  1. Define
  2. Measure
  3. Analyze
  4. Improve
  5. Control
- **DMADV** is used to create new product designs or process designs. The DMADV project methodology also has five phases:
  1. Define
  2. Measure
  3. Analyze
  4. Design
  5. Verify

Six Sigma **because the term sigma refers to one standard deviation in a data set.**