8.3
Software Testing, Cost Estimation, Quality Management and Configuration Management

Software Testing

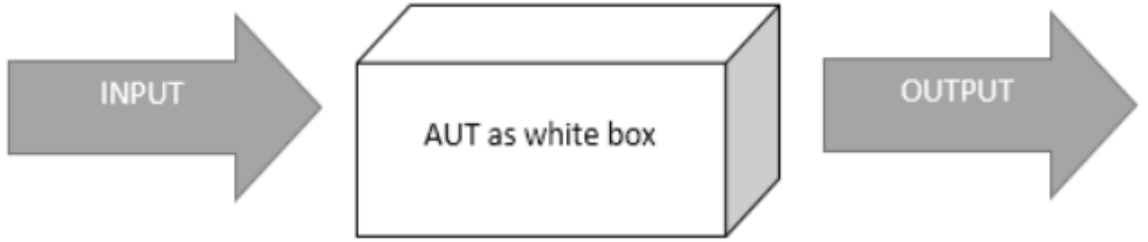
- Software testing is a process used to identify the correctness, completeness and quality of developed computer software.
- It is the process of executing a program / application under positive and negative conditions by manual or automated means.
- It checks for Specification, Functionality and Performance
- It is a validation and verification process.
- **Testing = Verification + Validation**
- Verification: Static testing (no run)
- Validation: Dynamic testing (code running)

Bug, Fault and Failure

"A person makes an **Error**
That creates a **fault** in the software
That can cause a **failure** in the operation"

- **Error:** An error is a human action that produces the incorrect result that results in a fault.
- **Bug:** The presence of error at the time of execution of the software.
- **Fault:** State of software caused by an error.
- **Failure:** Deviation of the software from its expected result. It is an event.
- **Test data:** A test data is the inputs which have been devised to test the system.
- **Test Case:** A test case is the triplet [I, S, O], where I is the data input to the system, S is the state of the system at which data is input, and O is the expected output of the system or Inputs to test the system and the predicted outputs from these inputs if the system operates according to its specification.

Techniques of Testing:



White-box testing:

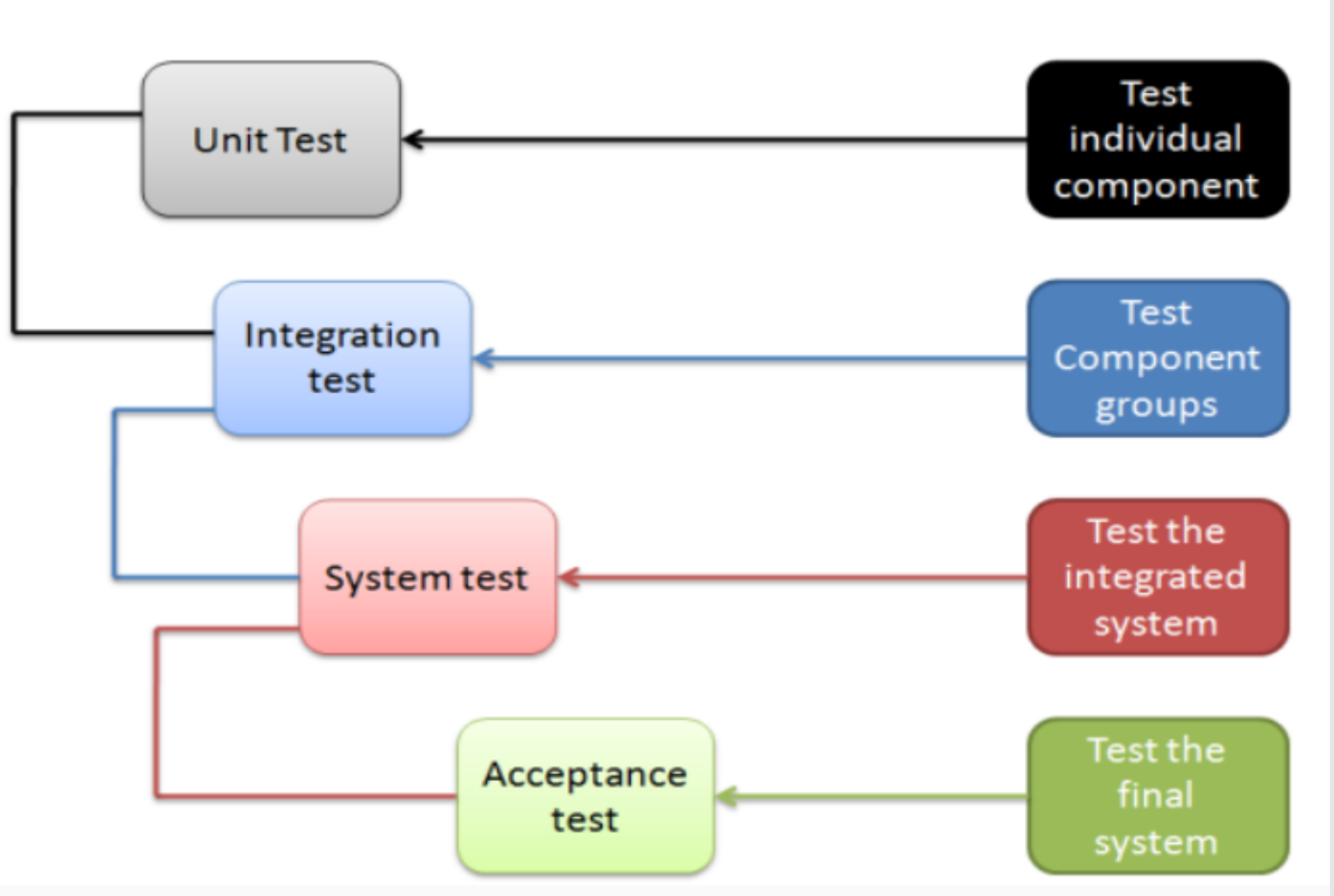
- It is a software testing method in which the internal structure of the item being tested is known to the tester.
- It is often used for verification and is done by the Developers.
- Also called Open testing, Clear-box testing, Code-based testing, Glass-box testing, Logic-coverage testing, Logic-driven testing, Structural testing, Structure-based testing, etc.
- **Example:** A Car mechanic should know the internal structure of the car engine to repair it.
  - In this example,
  - **CAR** is the **AUT (Application under Test).**
  - **User** is the **black box tester.**
  - **Mechanic** is the **white box tester.**

Techniques of Testing:

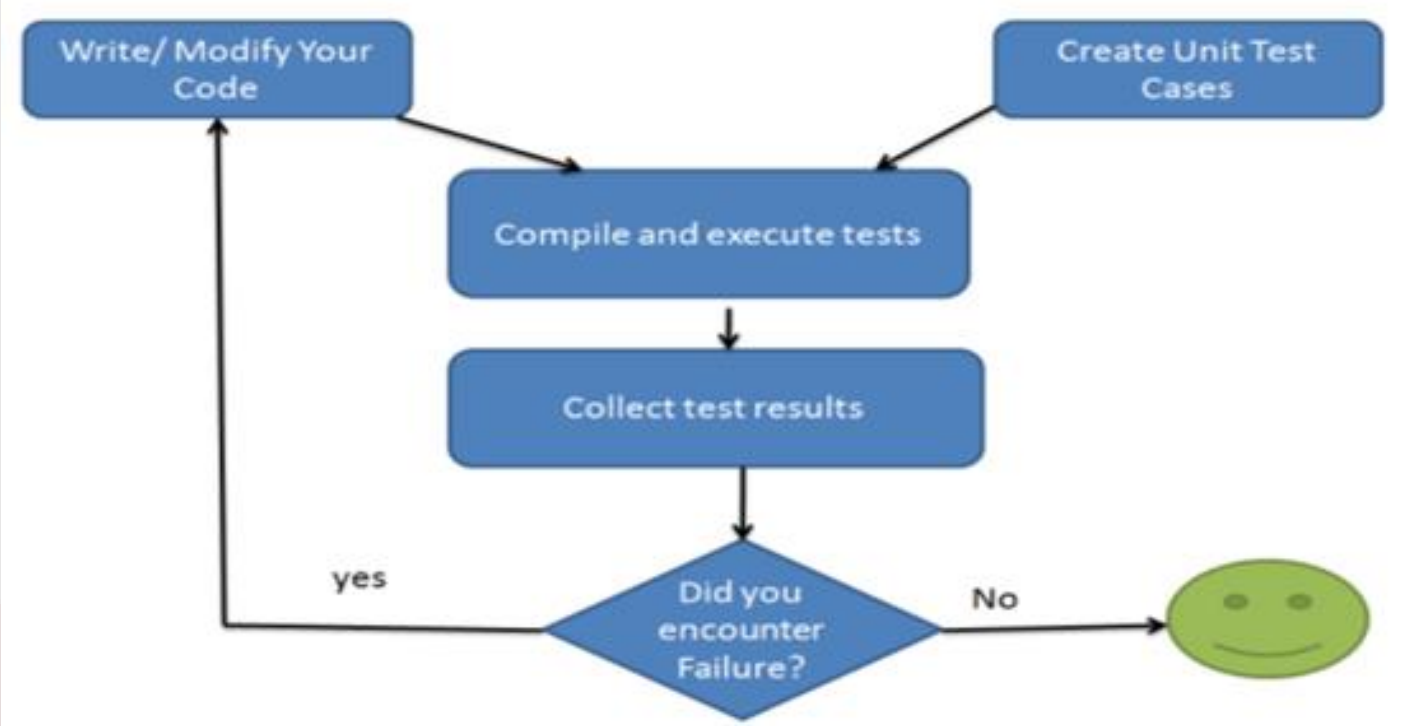
Blackbox Testing:

- It is a software testing method in which the internal structure of the item being tested is not known to the tester.
- In it, testing will be done by visualizing the application as a black box.
- It is used for Validation and is done by the professional testing team.
- Also called Specification-Based Testing, Closed Testing.
- **Example:** A simple example of black-box testing is a TV (Television). As a user, we watch the TV but we don't need the knowledge of how the TV is built and how it works, etc. We just need to know how to operate the remote control to switch on, switch off, change channels, increase/decrease volume, etc.
- In this example,
  - The **TV** is your **AUT (Application under Test).**
  - The **remote control** is the User Interface (UI) that you use to test.
  - You just need to know how to use the application.

Levels of Testing

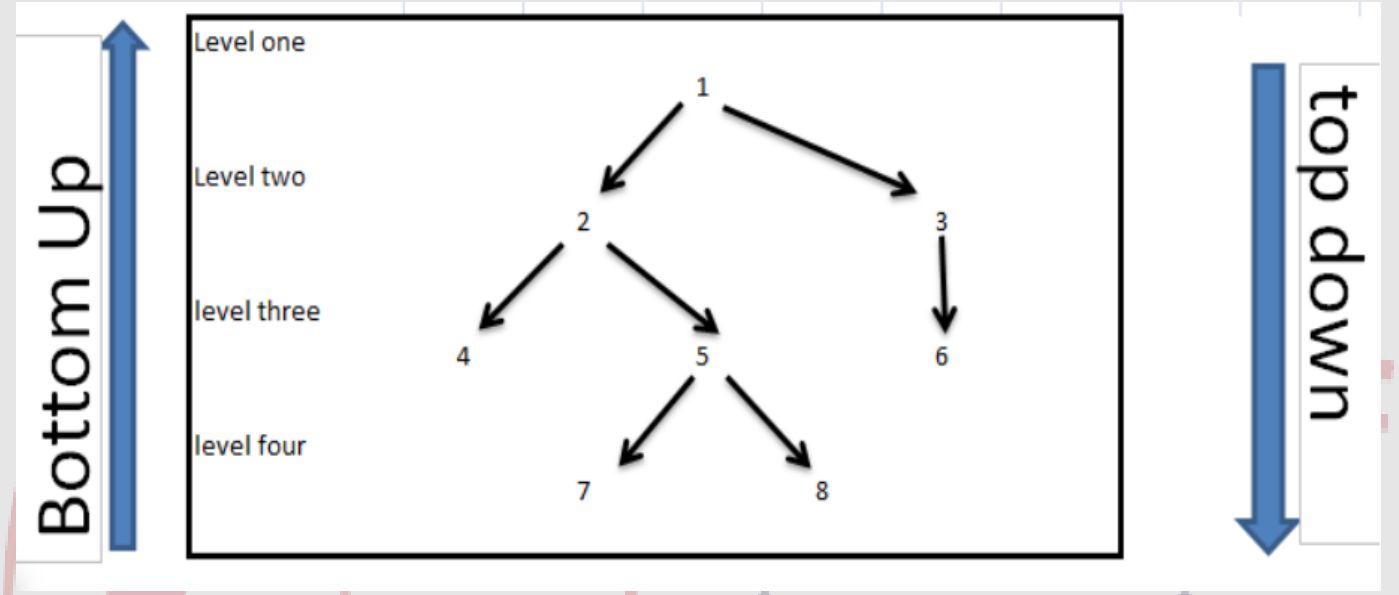


Unit Testing



- Unit testing is a method by which individual units of source code together are tested to determine if they are fit for use.
- The smallest independent and testable part of the source code is referred to as a unit. It could be a function, a file or a program.
- Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended.

Integration Testing



- It is done after unit testing.
- In it, all the units are integrated together and tested as a group to make sure that integrated system is ready for system testing.
- It checks the data flow from one module to other modules. This kind of testing is performed by testers.

Types of Integration Testing

i. Big bang testing:

- As the name suggests, in big bang form of testing all the modules are combined to form a complete system and then tested for bugs.

ii. Top down testing:

- It is a method in which integration testing takes place from top to bottom following the control flow of software system.
- The higher level modules are tested first and then lower level modules are tested and integrated in order to check the software functionality.

iii. Bottom up testing:

- It is a method in which the lower level modules are tested first.
- These tested modules are then further used to facilitate the testing of higher level modules. The process continues until all modules at top level are tested. Once the lower level modules are tested and integrated, then the next level of modules are formed.

System Testing

- It is a level of software testing where a complete, integrated system is tested.
- The purpose of this test is to evaluate system's compliance with the specified requirements.
- After integration testing, fully integrated application is tested to check that where the system meets its software requirements specification (SRS).
- Falls under black box testing
- It is carried out to check the behavior of the application, software design and expectation of the end user.

Types of System Testing

i. Usability Testing:

- To make sure that the system is easy to use, learn and operate.

ii. Load Testing:

- Is necessary to know that a software solution will perform under real-life loads

iii. Regression Testing:

- Involves testing done to make sure none of the changes made over the course of the development process have caused new bugs.
- It also makes sure no old bugs appear from the addition of new software modules over time.

iv. Recovery Testing:

- To make sure how well the system recovers from various input errors and other failure situations.

v. Migration Testing:

- Is done to ensure that the software can be moved from older system infrastructures to current system infrastructures without any issues.

Types of System Testing

vi. Functional Testing:

- It involves trying to think of any possible missing functions. Testers might make a list of additional functionalities that a product could have to improve it during functional testing.

vii. Documentation Testing:

- To make sure that the system's user guide and other help topics documents are correct and usable.

viii. Interoperability Testing:

- To make sure whether the system can operate well with third-party products or not.

ix. Performance Testing:

- To make sure the system's performance under the various condition, in terms of performance characteristics.

x. Scalability Testing:

- To make sure the system's scaling abilities in various terms like user scaling, geographic scaling, and resource scaling.

xi. Reliability Testing:

- To make sure the system can be operated for a longer duration without developing failures.

Acceptance Testing

- It is a test conducted to find if the requirements of a specification or contract are met as per its delivery.
- It is basically done by the user or customer. However, other stockholders can be involved in this process.

Types of Acceptance Testing

Alpha Testing:

- It is a type of acceptance testing; performed to identify all possible issues and bugs before releasing the final product to the end users.
- Alpha testing is carried out by the testers who are internal employees of the organization.
- The main goal is to identify the tasks that a typical user might perform and test them.

Advantages of Alpha Testing:

- Provides better view about the reliability of the software at an early stage
- Helps simulate real time user behavior and environment.
- Detect many showstopper or serious errors
- Ability to provide early detection of errors with respect to design and functionality.

Disadvantages of Alpha Testing:

- In depth, functionality cannot be tested as software is still under development stage. Sometimes developers and testers are dissatisfied with the results of alpha testing.

Beta Testing:

- **It** is performed by “real users” of the software application in “real environment” and it can be considered as a form of external user acceptance testing.
- It is the final test before shipping a product to the customers. Direct feedback from customers is a major advantage of Beta Testing. This testing helps to test products in customer’s environment.
- Beta version of the software is released to a limited number of end-users of the product to obtain feedback on the product quality. Beta testing reduces product failure risks and provides increased quality of the product through customer validation.

Advantages of Beta Testing

- Reduces product failure risk via customer validation.
- Beta Testing allows a company to test post-launch infrastructure.
- Improves product quality via customer feedback
- Cost effective compared to similar data gathering methods
- Creates goodwill with customers and increases customer satisfaction.

Disadvantages of Beta Testing

- Test Management is an issue. As compared to other testing types which are usually executed inside a company in a controlled environment, beta testing is executed out in the real world where you seldom have control.
- Finding the right beta users and maintaining their participation could be a challenge

Test Case Design

- It involves creating a set of test cases that can be used to verify the functionality and performance of a software application.
- The goal of test case design is to ensure that the software meets its specified requirements and that it is reliable and error-free.

Design Approaches:

a. Requirements-based testing:

- In it, test cases are designed based on test objectives and test conditions that are derived from requirements.
- It includes functional tests and also non-functional attributes such as performance, reliability or usability.

b. Partition Testing:

- It is a software testing technique that divides the input data of a software unit into partitions of equivalent data from which test cases can be derived.
- This technique tries to define test cases that uncover classes of errors, thereby reducing the total number of test cases that must be developed.

c. Structural Testing[white box testing]

Sample Test case

Test Scenario ID		**Test Case ID**	
Test Case Description		**Test Priority**	
Pre-Requisite		**Post-Requisite**	

Test Execution Steps:

S.No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments

Negative Test Cases for Login

Test Scenario ID	Login-1	Test Case ID	Login-1B
Test Case Description	Login – Negative test case	Test Priority	High
Pre-Requisite	NA	Post-Requisite	NA

Test Execution Steps:

S.No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments
1	Launch application	https://www.facebook.com/	Facebook home	Facebook home	IE-11	Pass	[Priya 10/17/2017 11:44 AM]: Launch successful
2	Enter invalid Email & any Password and hit login button	Email id : invalid@xyz.com Password: ******	The email address or phone number that you've entered doesn't match any account. Sign up for an account.	The email address or phone number that you've entered doesn't match any account. Sign up for an account.	IE-11	Pass	[Priya 10/17/2017 11:45 AM]: Invalid login attempt stopped
3	Enter valid Email & incorrect Password and hit login button	Email id : valid@xyz.com Password: ******	The password that you've entered is incorrect. Forgotten password?	The password that you've entered is incorrect. Forgotten password ?	IE-11	Pass	[Priya 10/17/2017 11:46 AM]: Invalid login attempt stopped

Positive Test Cases for Login

Test Scenario ID	Login-1	Test Case ID	Login-1A
Test Case Description	Login – Positive test case	Test Priority	High
Pre-Requisite	A valid user account	Post-Requisite	NA

Test Execution Steps:

S.No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments
1	Launch application	https://www.facebook.com/	Facebook home	Facebook home	IE-11	Pass	[Priya 10/17/2017 11:44 AM]: Launch successful
2	Enter correct Email & Password and hit login button	Email id : test@xyz.com Password: ******	Login success	Login success	IE-11	Pass	[Priya 10/17/2017 11:45 AM]: Login successful

Test Automation

- In software testing, test automation is the use of special software (separate from the software being tested) to control the execution of tests and the comparison of actual outcomes to predicted outcomes.
- Test automation can automate previous repetitive but necessary testing in a formalized testing process already in place, or add additional testing that would be difficult to perform manually.
- It reduces testing costs by supporting the test process with range of software tools.
- There are two general approaches to test automation:
- **Code-driven testing**. It is a software development approach in which it uses testing frameworks that allows the execution of unit tests to determine whether various sections of the code are acting accordingly as expected under various conditions.
- **Graphical user interface testing**. A testing framework generates user interface events such as keystrokes and mouse clicks, and observes the changes that result in the user interface, to validate that the observable behavior of the program is correct.

Algorithmic cost modelling

- It is a technique used in software engineering to estimate the cost of developing software algorithms.
- This approach involves analyzing the algorithm and breaking it down into smaller components, such as the number of input and output variables, the complexity of the logic, and the amount of computation required.
- It can be built by analyzing the costs and attributes of completed projects and finding the closest fit formula to actual experience.
- Effort= $A^* \text{ size}^B * M$

  Where; A is an organization-dependent constant

  B reflects the disproportionate effort for large project

  M is a multiplier reflecting product, process and people attributes

Software Quality Assurance

- Software quality assurance (SQA) aims at ensuring that the final product will have an acceptable quality.
- It is a planned and systematic plan of all actions necessary to provide adequate confidence that an item or product conforms to establish technical requirements.
- It is mainly a management activity to identify quality problems early in the development.
- Quality assurance focuses on both the product and the process.

Software quality management

- Concerned with ensuring that the required level of quality is achieved in a software product
- Involves defining appropriate quality standards and procedures and ensuring that these are followed
- Should aim to develop a ‘quality culture’ where quality is seen as everyone’s responsibility

Quality Management Activities

i. Quality Assurance:

- Establishes a framework of organizational procedures and standards for high quality s/w.
- Consists of auditing and reporting procedures used to provide management with data needed to make proactive decisions.
- If data provided through QA identifies problems, management is responsible to address the problem and apply the necessary resources to resolve quality issues.

Quality Management Activities

ii. Quality Planning:

- Selection of appropriate procedures and standards from this framework and adaptation of these for specific s/w projects and modify these as required.
- A quality plan sets out the desired product qualities and how these are assessed and the most significant quality attributes.

iii. Quality Control:

- It ensures that the project quality procedures and standards are followed by the s/w development team.
- QC involves the series of inspection, reviews and test used throughout the s/w process to ensure conformance of a work product to its specifications.

Formal Technical Review

- Formal technical review (FTR) is a software quality control activity performed by software engineers and others.
- The objectives of an FTR are:
  - To uncover errors in function, logic, or implementation for any representation of the software;
  - To verify that the software under review meets its requirements;
  - To ensure that the software has been represented according to predefined standards
  - To achieve software that is developed in a uniform manner;
  - To make projects more manageable.

Steps in Formal Technical Review

1. **The review meeting:** Each review meeting should be held considering the following constraints

- Involvement of people *should be b*etween 3, 4 and 5 people.
- Advance preparation should occur but it should be very short that is at the most 2 hours of work for every person.
- The short duration of the review meeting should be less than two hour. Gives these constraints, it should be clear that an FTR focuses on specific (and small) part of the overall software.
- At the end of the review, all attendees of FTR must decide what to do.

Steps in Formal Technical Review

2. **Review reporting and record keeping :-**

- During the FTR, the reviewer actively records all issues that have been raised.
- At the end of the meeting all these issues raised are consolidated and a review list is prepared.
- Finally, a formal technical review summary report is prepared.
- It answers three questions :-
  - What was reviewed ?
  - Who reviewed it ?
  - What were the findings and conclusions ?

Steps in Formal Technical Review

3. **Review guidelines :-** Guidelines for the conducting of formal technical reviews should be established in advance. These guidelines must be distributed to all reviewers, agreed upon, and then followed. A review that is unregistered can often be worse than a review that does not minimum set of guidelines for FTR.

- Review the product, not the manufacture (producer).
- Take written notes (record purpose)
- Limit the number of participants and insists upon advance preparation.
- Develop a checklist for each product that is likely to be reviewed.
- Allocate resources and time schedule for FTRs in order to maintain time schedule.
- Conduct meaningful training for all reviewers in order to make reviews effective.
- Reviews earlier reviews which serve as the base for the current review being conducted.
- Set an agenda and maintain it.
- Separate the problem areas, but do not attempt to solve every problem notes.

Formal approaches to SQA

- Over the past few years, software engineering has argues that SQA needs more formal approach.
- Approaches:
  - Proof of correctness
  - Statistical SQA
  - Cleanroom process combines item 1 and 2

Proof of correctness

- It is a formal approach in SQA that involves using mathematical and logical methods to demonstrate that a software system satisfies its specifications or requirements.
- The main goal is to rigorously prove that the software behaves as intended, and it is free from design and implementation errors.

Statistical SQA

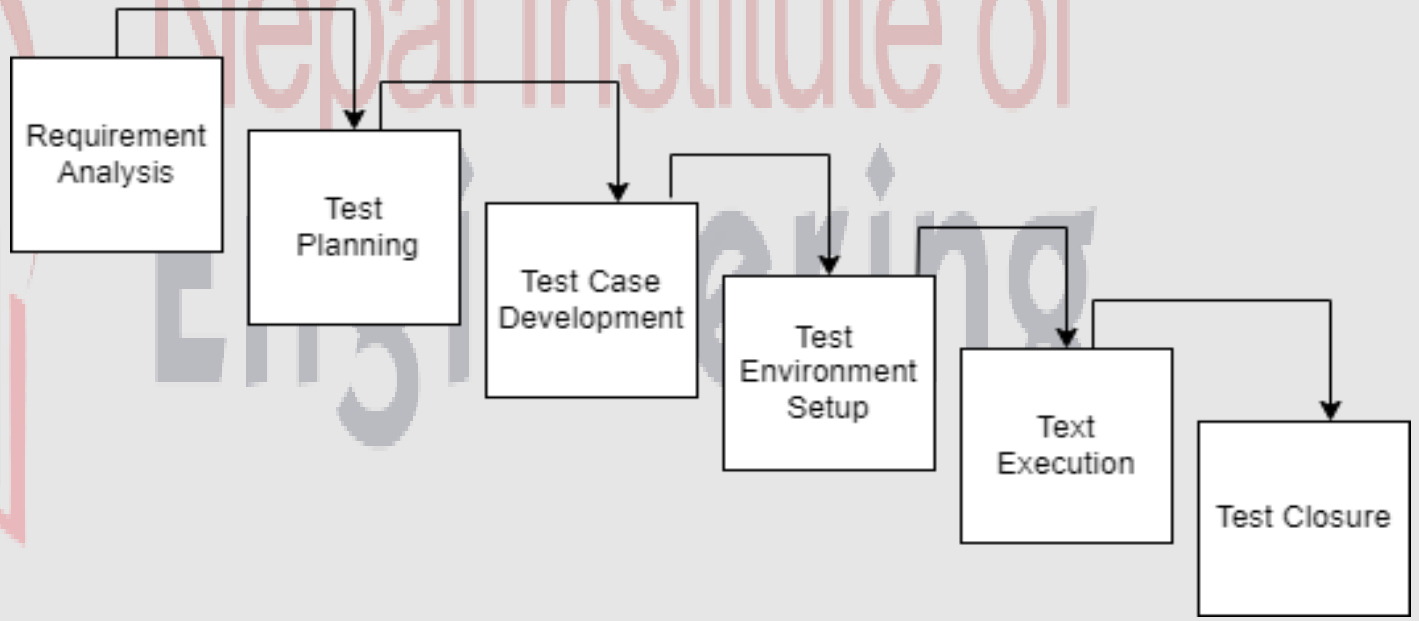
- It uses statistical methods to assess the quality of software. This can involve collecting data on the number of defects in the software, the severity of the defects, and the time it takes to fix the defects.
- Statistical SQA can be used to identify trends in the quality of software and to make predictions about the future quality of the software.

Cleanroom process

- It is a software development process that combines proof of correctness and statistical SQA.
- The Cleanroom process starts with the construction of a formal model of the software.
- The model is then used to generate test cases that are used to verify the correctness of the software.
- The Cleanroom process also uses statistical methods to assess the quality of the software.

Software Testing Life Cycle

- **Software Testing Life Cycle (STLC)** is the testing process which is executed in systematic and planned manner. In STLC process, different activities are carried out to improve the quality of the product.



Test Development Cycle Steps

1. Requirement Analysis:

- Requirement Analysis is the first step of Software Testing Life Cycle (STLC).
- In this phase quality assurance team understands the requirements like what is to be tested.
- If anything is missing or not understandable then quality assurance team meets with the stakeholders to better understand the detail knowledge of requirement.

2. Test Planning:

- Test Planning is most efficient phase of software testing life cycle where all testing plans are defined.
- In this phase manager of the testing team calculates estimated effort and cost for the testing work. This phase gets started once the requirement gathering phase is completed.

Test Development Cycle Steps

3. Test Case Development:

- The test case development phase gets started once the test planning phase is completed. In this phase testing team note down the detailed test cases.
- Testing team also prepare the required test data for the testing. When the test cases are prepared then they are reviewed by quality assurance team.

4. Test Environment Setup:

- Test environment setup is the vital part of the STLC. Basically test environment decides the conditions on which software is tested.
- This is independent activity and can be started along with test case development. In this process the testing team is not involved either the developer or the customer creates the testing environment.

Test Development Cycle Steps

5. Test Execution:

- After the test case development and test environment setup test execution phase gets started.
- In this phase testing team start executing test cases based on prepared test cases in the earlier step.

6. Test Closure:

- This is the last stage of STLC in which the process of testing is analyzed.

A framework for software metrics

- Software metric is a measure of some property of a piece of software or its specifications. Good quality, reliability and maintainability are important attributes of enterprise applications and have a huge impact on the success on the economics of the businesses powered. It really, really matters.

A Framework for Software Metrics: Measures, Metrics, and Indicators. These three terms are often used interchangeably, but they can have subtle differences

- Measure :Provides a quantitative indication of the extent, amount, dimension, capacity, or size of some attribute of a product or process
- Measurement: The act of determining a measure
- Metric:(IEEE) A quantitative measure of the degree to which a system, component, or process possesses a given attribute
- Indicator: A metric or combination of metrics that provides insight into the software process, a software project, or the product itself

Metrics for analysis and design model

Metrics for the analysis

- Functionality delivered
  - Provides an indirect measure of the functionality that is packaged within the software
- System size
  - Measures the overall size of the system defined in terms of information available as part of the analysis model
- Specification quality
  - Provides an indication of the specificity and completeness of a requirements specification

Metrics for the Design Model

- Architectural metrics
  - Provide an indication of the quality of the architectural design
- Component-level metrics
  - Measure the complexity of software components and other characteristics that have a bearing on quality
- Interface design metrics
  - Focus primarily on usability
- Specialized object-oriented design metrics
  - Measure characteristics of classes and their communication and collaboration characteristics

Configuration management

- Configuration management (CM) is the management of system change to software products.
- CM is the art of identifying, organizing, and controlling modifications to the software being built by a programming team. The goal is to maximize productivity by minimizing mistakes.
- SCM activities are developed to:
  - Identify the change
  - Control change
  - Ensure that change is being properly implemented
  - Report changes to others who may have an interest

Configuration management planning

- Configuration management (CM) is concerned with the policies, processes, and tools for managing changing software systems.
- CM planning defines:
  - the types of documents to be managed and a document scheme.
  - who takes responsibility for the CM procedures and creation baseline
  - policies for the change control and version management.
  - the tools which should be used to assist the CM process and only limitation on their use.
  - the CM database used to record configuration information

Change Management

- It is a critical process in software engineering that involves managing changes to software products or systems, while ensuring that they are executed in a controlled and efficient manner.
- Its goal is to ensure that any changes made to a software product are implemented successfully, without introducing negative impacts or causing disruptions to the overall system.
- It is the process of analyzing the costs and benefits of proposed changes, approving those changes that are cost-effective, and tracking which components in the system have been changed.

Versions/variants/releases

- **Version** An instance of a system which is functionally distinct in some way from other system instances.
- **Variant** An instance of a system which is functionally identical but non-functionally distinct from other instances of a system.
- **Release** An instance of a system which is distributed to users outside of the development team.

Version identification

- It helps to ensure that different versions of a software product are properly tracked, documented, and managed throughout their lifecycle, and enables developers and other stakeholders to easily communicate about different versions of the software.

Release Management

- It is a process in software engineering that involves managing the planning, scheduling, testing, and deployment of software releases.
- Its goal is to ensure that software releases are delivered on time, with the required quality, and without causing disruption to the organization.
- Effective release management requires collaboration among various stakeholders, including developers, project managers, quality assurance professionals, and end-users.
- It also requires the use of tools and techniques that help to streamline the process of managing software releases, such as version control systems, build automation tools, and continuous integration/continuous deployment (CI/CD) pipelines.

ISO (International Organization for Standardization) Standards

- Before ISO Standards, organizations had their own rules for making and testing software.
- Using different standards led to issues in maintaining software quality.
- New standards were created to improve software quality and testing techniques.
- These standards provide rules and procedures to enhance software and user experience.
- ISO aims to promote standardization and related activities globally.
- Its mission is to ease the international exchange of products and services.

International Standard for Software Testing

- **ISO/IEC/IEEE:** International Organization for Standardization and the International Electrotechnical Commission and Institute of Electrical and Electronics Engineers
- The standard having the number 29119 is developed for maintaining the correct software testing procedures for the software development.
- ISO/IEC/IEEE Standard 29119 is a collection of standards for software testing of any SDLC phases for any organization.
- For increasing the International Quality Assurance, we need to implement this standard in our software testing process, only then we will get internationally recognized product.

ISO/IEC/IEEE 29119

ISO/IEC/IEEE 29119 consists of five standards for international software testing standards which are:

- **ISO/IEC 29119-1**: This standard gives the concepts and meanings of software which are very useful in software development processes. This standard was published in September 2013.
- **ISO/IEC 29119-2**: This standard is also sub part of ISO Standard 29119 which deals with all the test processes for a better product output. This standard was also published in September 2013.
- **ISO/IEC 29119-3**: This standard has other importance related to the documentation of the product; therefore, it is responsible for delivering the complete documentation of the product. This standard is also published with previous 2 standards just discussed above, in September 2013.
- **ISO/IEC 29119-4:** This standard gives the right testing techniques and strategies for doing the software testing. This standard was published in 2014.
- **ISO/IEC 29119-5**: ISO 29119-5 has some unique importance that deals with keyword-based software testing which means that a keyword is used in complete testing and obtained the results, the keyword can be any word which can give better testing results. This Standard was published in 2015.

ISO/IEC 9241-11:

- This standard deals with the customer or users' satisfaction of the product and the effectiveness and efficiency of the product is also described in this standard.
- This standard must be included in any software development process for achieving best user experience and increasing the efficiency and effectiveness of the product.

ISO/IEC 25000-2005

- It provides the procedures and rules for achieving the best software quality requirements and evaluation (SQuaRE) of the software.
- SQuaRE is an abbreviation used for Software Quality Requirements and Evaluation, has some sub-parts of the standard which are:
- **ISO 2500N** – This standard is used for managing the quality divisions.
- **ISO 2501N** – This standard deal with Quality Model Division of the software development.
- **ISO 2502N** – ISO 2502N standard consist of Quality Measurement Division which are also related to Software Quality metrics.
- **ISO 2503N** – This ISO standard is useful for the divisions of software quality requirements.
- **ISO 2504N** – This standard deal with Quality Evaluation Division of the software.

ISO/IEC 9126:

- This standard is also useful for determining the Quality of the Software product which includes – quality model of the software, the metrics related to the quality, the internal and external metrics of the software application.
- This standard is a popular standard because of its usage and importance in software testing.
- The quality attributes considered in this standard are
  - maintainability,
  - usability,
  - portability and
  - functionality.
- The other attributes can also be included which are
  - user's reliability of the software,
  - the right working of the product and
  - high efficiency

ISO/IEC 12119:

- This ISO standard is responsible for the software packages which are to be delivered to customer or users.
- The main aim of this Standard is to deliver the quality software package as possible for the customer, but this standard doesn't deal with customer or client production process, rather it deals with software package production processes.

CMMI

- The Capability Maturity Model Integration (CMMI) is a process and behavioral model that helps organizations streamline process improvement and encourage productive, efficient behaviors that decrease risks in software, product, and service development.
- **Objectives of CMMI:**
  - Fulfilling customer needs and expectations.
  - Value creation for investors/stockholders.
  - Market growth is increased.
  - Improved quality of products and services.
  - Enhanced reputation in Industry.

CMMI Model – Maturity Levels:

- **Maturity level 1 : Initial**
  - processes are poorly managed or controlled.
  - unpredictable outcomes of processes involved.
  - chaotic approach used.
  - No KPAs (Key Process Areas) defined.
  - Lowest quality and highest risk.

CMMI Model – Maturity Levels:

- **Maturity level 2 : Managed**
  - requirements are managed.
  - processes are planned and controlled.
  - projects are managed and implemented according to their documented plans.
  - This risk involved is lower than Initial level, but still exists.
  - Quality is better than Initial level.

CMMI Model – Maturity Levels:

- **Maturity level 3 : Defined**
  - processes are well characterized and described using standards, proper procedures, and methods, tools, etc.
  - Medium quality and medium risk involved.
  - Focus is process standardization.

CMMI Model – Maturity Levels:

- **Maturity level 4 : Quantitatively managed**
  - quantitative objectives for process performance and quality are set.
  - quantitative objectives are based on customer requirements, organization needs, etc.
  - process performance measures are analyzed quantitatively.
  - higher quality of processes is achieved.
  - lower risk

CMMI Model – Maturity Levels:

- **Maturity level 5 : Optimizing**
  - continuous improvement in processes and their performance.
  - improvement has to be both incremental and innovative.
  - highest quality of processes.
  - lowest risk in processes and their performance.


THANK
YOU