

ISTQB CTFL

Fundamentals of Testing

What is Testing

- *Process consisting of all life cycle activities both static and dynamic, concerned with Planning, preparation and evaluation of software products and related work products to determine that they satisfy specified requirements and to demonstrate that they are fit for purpose and to detect defects.*
- Testing is more than running tests
- Testing is more than verification

Verification

VS

Validation

- ***Confirmation by examination and through provision of objective evidence that the specified requirements have been fulfilled***
- Verification is checking against the specification
- Concerned with evaluating a work product, component or system to determine whether it meets specified requirements

- ***Confirmation by examination and through provision of objective evidence that the requirements for a specific intended use or application have been fulfilled***
- Validation is checking if we have built right system
- Concerned with evaluating a work product, component or system to determine whether it meets the user needs and requirements

Test Objective

- *Purpose for testing*
- *Common test objectives are below*
- To evaluate the work products such as requirements, user stories, design and code by using static testing technique such as reviews - **Static testing**
- To verify whether all specified requirements have been fulfilled - **User Acceptance Testing**
- To validate whether the test object is complete and works as the user and other stakeholders expect- **User Acceptance Testing**
- To build confidence in the level of quality of the test object, such as tests with high risk pass failure that are observed are acceptable
- To prevent defects by early test activities such as reviews and early test design - **Test Design**
- To find failures and defect - **Test Execution**
- To provide sufficient information to stakeholders to allow them to make informed decisions, especially regarding level of quality of the test object (Satisfying entry and exit criteria) - **Test monitoring**
- To reduce the level of risk of inadequate software quality (previously undetected failures occurring in operation) - **Regression**
- To comply with contractual, legal or regulatory standards or requirements and/or to verify the test object's compliance with such requirements or standards

Test Object

- *Component or system to be tested*

Testing

VS Debugging

- Finding failures caused by defects
- Done by testing team

- ***Process of finding, analyzing and removing the causes of failures in component or system***
- A development activity where member of team, finds analyses and fixes the defect
- Usually done by development team

Testing's contribution to success

- Early involvement of testers in requirement reviews and user story refinement could detect defects in these work products before design or coding which can prevent the design or dev of wrong product
- Testers closely working with dev during system design will increase each party's understanding of the design and how to test it
- Testers closely working with dev during coding will increase each party's understanding of code and how to test it
- Having testers verify and validate the software prior to release can detect failures that might otherwise have been missed
- Achieving defined test objectives contributes to overall software development and maintenance

Quality Control

- ***Activities designed to evaluate the quality of a component or system***
- Concerned with quality of products rather than processes, to ensure that they have achieved the desired level or quality
- Testing is quality control activity

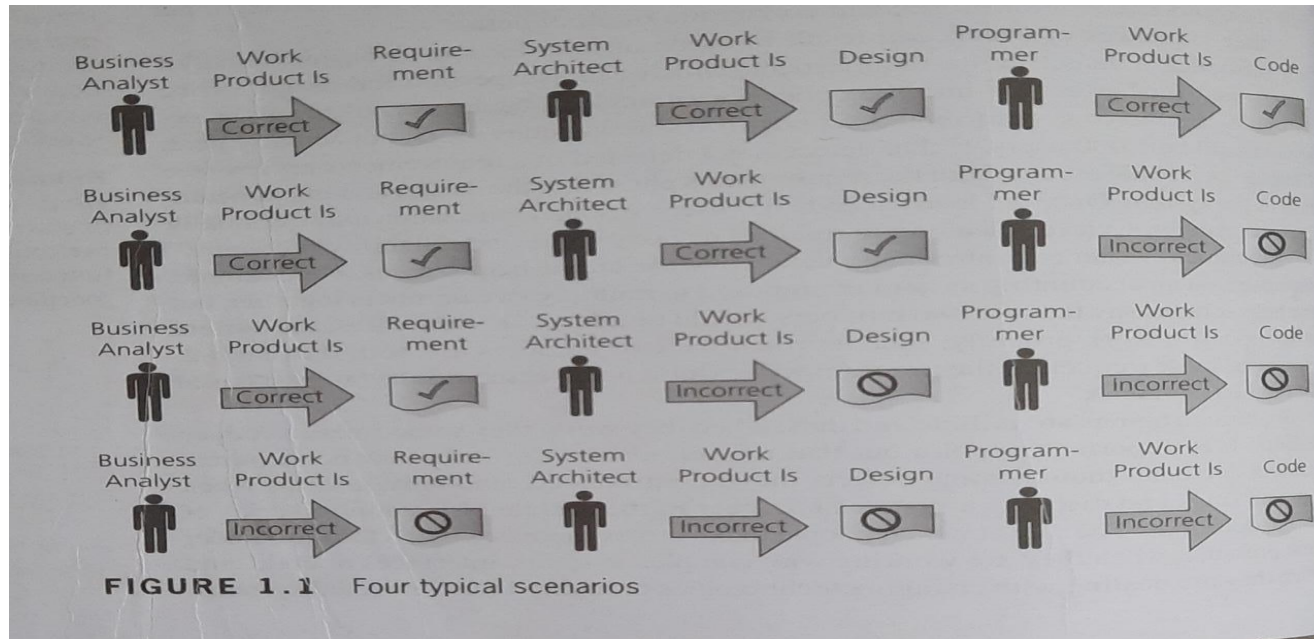
Quality Assurance

- ***Activities focused on providing confidence that quality requirements will be fulfilled***
- Associated with ensuring that a company's standard ways of performing various tasks are carried out correctly
- Root cause analysis and retrospectives are used to help to improve processes for more effective quality assurance

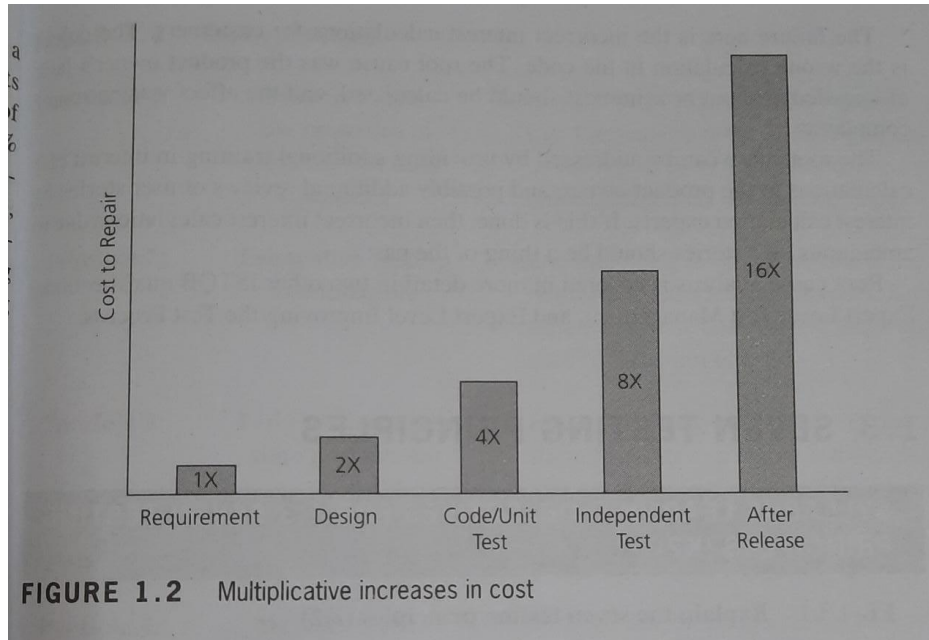
Error, Defect, Failure, Root cause

- *Error or mistake is a human action that produces an incorrect result*
- *Defect (bug, fault) is An imperfection or deficiency in a work product where it does not meet its requirements or specifications*
- *Failure is An event in which a component or system does not perform a required function within specified limits*
- *Root cause is A source of a defect such that if it is removed, the occurrence of the defect type is decreased or removed*

Introduction of defects in different phases



Cost impact of introducing defects



Testing Principles

- Testing shows presence of defects, not the absence
- Exhaustive testing is impossible
- Early testing saves time and money
- Defects cluster together
- Beware of the pesticide paradox
- Testing is context dependent
- Absence of defect fallacy

Test Process in context

- Factors that influence test process
 - SDLC model and project methodologies used
 - Test levels and test types considered
 - Product and Project risks
 - Business domain (mobile apps vs medical devices)
 - Operational constraints
 - Budget and resources
 - Timescales
 - Complexity
 - Contractual and regulatory requirements
 - Organizational policies and practices
 - Required internal and external standards

Coverage , Test basis, KPI, Test Strategy

- ***Coverage is The degree to which specified coverage items are exercised by a test suite, expressed as a percentage***
- ***Test basis is The body of knowledge used as the basis for test analysis and design.***
 - *Whatever the tests are being derived from (requirements, user story, design or code)*
- ***KPI - Key Performance Indicator - indicator to measure progress***
- ***Test Strategy is A description of how to perform testing to reach test objectives under given circumstances***

Test activities and tasks

- Test Planning
- Test Monitoring and Control
- Test Analysis
- Test Design
- Test Implementation
- Test execution
- Test Closure

Test Planning

- *Test Planning is the activity of establishing or updating a test plan*
- *Test Plan is Documentation describing the test objectives to be achieved and the means and the schedule for achieving them, organized to coordinate testing activities*

Test Monitoring and Control

- *Test Monitoring is a test management activity that checks the status of testing activities, identifies any variances from planned or expected, and reports status to stakeholders*
- *Test control is a test management task that develops and applies corrective actions to get a test project on track when it deviates from what was planned*

Test Analysis

- ***Test Analysis is an activity that identifies test conditions by analyzing the test basis***
- ***Test condition (charter) is A testable aspect of a system or component identified as basis of testing***
- In test analysis, we analyze test basis to identify testable features and define associated test conditions
- Test analysis determines what to test
- Test basis can include requirements, user stories, design specifications, risk analysis reports, system design, architecture, interface specifications and user expectations
- In test analysis, we transform more general testing objectives defined in test plan into tangible test conditions

Activities of test analysis

- Analyze the test basis appropriate to the test level being considered
 - Requirement specification, ex - business requirements, system requirements, user stories epics etc say what the component or system should do and are the source of tests to assess functionality as well as non functional aspects
 - Design and implementation information such as architecture diagram, design specification, call flows , modelling diagram (UML, entity relationship) interface diagram etc can be useful source of coverage criteria to ensure sufficient testing has been done on those structures
 - The implementation itself including code, database metadata and queries, interfaces etc to identify what to be tested
 - Risk analysis reports to find out the areas where highest risk is identified which should have more tests

Activities of test analysis

- Evaluate the test basis and test items to identify various types of defects that might occur
 - Ambiguities
 - Omissions
 - Inconsistencies
 - Inaccuracies
 - Contradictions
 - Superfluous statements
- Identify features and sets of features to be tested
- Identify and prioritize test conditions for each feature, based on analysis of the test basis, and considering functional non functional and structural characteristics other business and technical factors and level of risks
- Capture bi-directional traceability between each element of test basis and associated test conditions

Test Design

- ***The activity that derives and specifies test cases from test conditions***
- ***Test Design includes***
 - *Design and prioritize test cases and sets of test cases*
 - *Identify the necessary test data to support the test conditions and test cases as they are identified and designed*
 - *Design the test environment, including setup, and identify any required infrastructure and tools*
 - *Capture bi-directional traceability between the test basis, test conditions, test cases and test procedure*
- ***Test Case is A set of preconditions, inputs, actions (where applicable), expected results and postconditions, developed based on test conditions***
- ***Test Data is Data needed for test execution***
 - *Data created or selected to satisfy the execution preconditions and inputs to execute one or more test cases*
- ***Test Procedure is A sequence of test cases in execution order, and any associated actions that may be required to set up the initial preconditions and any wrap up activities post execution***
- ***Test Suite is A set of test scripts or test procedures to be executed in a specific test run***

Test Implementation

- ***The activity that prepares the testware needed for test execution based on test analysis and design***
- Test implementation includes:
 - Develop and prioritize test procedures and potentially create automated test scripts
 - Create test suites from the test procedures and automated test scripts
 - Arrange test suites within the test execution schedule in a way that results in efficient test execution
 - Build test environment (possibly including test harness, service virtualization, simulators and other infrastructure items) and verify everything needed has been setup correctly
 - Prepare test data and ensure that it is properly loaded in the test environment (including inputs, data resident in databases and other data repositories and system configuration data)
 - Verify and update the bi-directional traceability between test basis, test conditions, test cases, test procedures and test suites

Test execution

- ***The activity that runs a test on a component or system producing actual results***
- Test Execution includes
 - Record identities and version of all the test items including tools and other testware
 - Execute tests either manually or using an automated test script according to planned sequence
 - Compare actual results with expected results to observe if there is deviation and record **anomalies**
 - Analyze anomalies in order to establish their likely causes.
 - Report defects bases on failures observed. Either software defects or Test defect found in testware
 - Log the outcome of test execution including anomalies, pass/fail and versions
 - As necessary repeat test activities when actions are taken to rectify the bugs example. Re-run test to confirm bug is fixed and run regression tests to verify if bug fix has not introduced new bugs
 - Verify and update bi-directional traceability between the test basis, test conditions, test cases, test procedures and test results

Test Completion

- ***The activity that makes testware available for later use, leaves test environments in a satisfactory condition and communicates the results of testing to relevant stakeholders***
- Test completion includes:
 - Check whether all defect reports are closed, entering change requests or product backlog items for any defects that remain unresolved at the end of test execution
 - Create test summary report to be communicated to relevant stakeholders
 - Finalize and archive the test environment, the test data, test infrastructure and other testware for later reuse
 - Hand over the testware to the maintenance teams, other project teams, and/or stakeholders who could benefit from its use
 - Analyze lessons learnt from completed test activities to determine changes needed for future iterations, releases or projects
 - Use the information gathered to improve test process maturity, especially as an input to test planning for future projects

Test Work products

- ***Work products produced during the test process for use in planning, designing, executing, evaluating and reporting on testing***
- Test Planning work products
 - Test Plan
- Test Monitoring and control work products
 - Test report
- Test analysis work products
 - Test Conditions
 - Test defect report found during analysis (sometimes)
- Test Design work products
 - Test cases and set of test cases
 - Test data
 - Test environment design
 - Identification of test tools and infra

Test work products

- Test Implementation work products
 - Test procedures and sequencing of those procedures
 - Test suites
 - Test execution schedule
 - Test scripts to run on tools
- Text Execution work products
 - Documentation of status of individual test cases or test procedures
 - Defect reports
 - Documentation about which test items, test object, test tools and testware were involved
 - Instead of giving number of tests passed failed, its better to provide traceability data to stakeholders
- Test completion work products
 - Test summary reports
 - Action items for improvement of subsequent projects or iterations
 - Change requests or product backlog items
 - Finalized testware

Traceability between test basis and test work products

- ***Traceability is The ability to establish explicit relationships between related work products or items within work products***
- Good traceability supports
 - Analyzing the impact of changes, whether to requirements or to the component or system
 - Making testing suitable and being able to measure coverage
 - Meeting IT governance criteria
 - Improving coherence of the test progress reports and test summary reports
 - Relating the technical aspects of testing to stakeholders in terms that they can understand
 - Providing information to assess product quality, process capability and project progress against business goals

Generic Skills required for testing

- Curiosity
- Professional pessimism
- Critical eye
- Attention to detail
- Experience
- Good Communication skill

Whole Team Approach

- ▶ The whole-team approach means involving everyone with the knowledge and skills necessary to ensure project success.
- ▶ The team includes representatives from the customer and other business stakeholders who determine product features.
- ▶ The team should be relatively small; successful teams have been observed with as few as three people and as many as nine.
- ▶ Ideally, the whole team shares the same workspace, as co-location strongly facilitates communication and interaction.
- ▶ The whole-team approach is supported through the daily stand-up meetings involving all members of the team, where work progress is communicated and any impediments to progress are highlighted.
- ▶ The whole-team approach promotes more effective and efficient team dynamics.

Benefits of whole team approach

- ▶ The use of a whole-team approach to product development is one of the main benefits of Agile development. Its benefits include:
 - ▶ Enhancing communication and collaboration within the team
 - ▶ Enabling the various skill sets within the team to be leveraged to the benefit of the project
 - ▶ Making quality everyone's responsibility

Benefits of whole team approach

- ▶ The whole team is responsible for quality in Agile projects.
- ▶ The essence of the whole-team approach lies in the testers, developers, and the business representatives working together in every step of the development process.
- ▶ Testers will work closely with both developers and business representatives to ensure that the desired quality levels are achieved.
- ▶ This includes supporting and collaborating with business representatives to help them create suitable acceptance tests, working with developers to agree on the testing strategy, and deciding on test automation approaches.
- ▶ Testers can thus transfer and extend testing knowledge to other team members and influence the development of the product.

Independence of Testing


Testing throughout SDLC

Impact of SDLC on testing

SDLC and Good testing practices

Test Driven development

Test Driven Development

- ▶ Test-driven development (TDD) is used to develop code guided by automated test cases.
 - ▶ The process for test-driven development is:
 - ▶ Add a test that captures the programmer's concept of the desired functioning of a small piece of code
 - ▶ Run the test, which should fail since the code doesn't exist
 - ▶ Write the code and run the test in a tight loop until the test passes
 - ▶ Refactor the code after the test is passed, re-running the test to ensure it continues to pass against the refactored code
 - ▶ Repeat this process for the next small piece of code, running the previous tests as well as the added tests
 - ▶ The tests written are primarily unit level and are code-focused, though tests may also be written at the integration or system levels.
- 


Acceptance Test Driven Development

Acceptance Test Driven Development

- ▶ Acceptance test-driven development defines acceptance criteria and tests during the creation of user stories.
- ▶ Acceptance test-driven development is a collaborative approach that allows every stakeholder to understand how the software component has to behave and what the developers, testers, and business representatives need to ensure this behavior.
- ▶ Acceptance test-driven development creates reusable tests for regression testing.
- ▶ Specific tools support creation and execution of such tests, often within the continuous integration process. These tools can connect to data and service layers of the application, which allows tests to be executed at the system or acceptance level.
- ▶ Acceptance test-driven development allows quick resolution of defects and validation of feature behavior. It helps determine if the acceptance criteria are met for the feature.

Behaviour Driven Development

Behavior Driven Development

- ▶ Behavior-driven development allows a developer to focus on testing the code based on the expected behavior of the software.
 - ▶ Because the tests are based on the exhibited behavior of the software, the tests are generally easier for other team members and stakeholders to understand.
 - ▶ Specific behavior-driven development frameworks can be used to define acceptance criteria based on the given/when/then format:
 - ▶ *Given* some initial context, *When* an event occurs, *Then* ensure some outcomes.
 - ▶ From these requirements, the behavior-driven development framework generates code that can be used by developers to create test cases.
 - ▶ Behavior-driven development helps the developer collaborate with other stakeholders, including testers, to define accurate unit tests focused on business needs
- 

DevOps and testing

Shift Left approach

Retrospectives and process improvement

- ▶ In Agile development, a retrospective is a meeting held at the end of each iteration to discuss what was successful, what could be improved, and how to incorporate the improvements and retain the successes in future iterations.
- ▶ Retrospectives cover topics such as the process, people, organizations, relationships, and tools.
- ▶ Regularly conducted retrospective meetings, when appropriate follow up activities occur, are critical to self-organization and continual improvement of development and testing.
- ▶ Retrospectives can result in test-related improvement decisions focused on test effectiveness, test productivity, test case quality, and team satisfaction. They may also address the testability of the applications, user stories, features, or system interfaces.
- ▶ The timing and organization of the retrospective depends on the particular Agile method followed.
- ▶ Testers should play an important role in the retrospectives. Testers are part of the team and bring their unique perspective.

Test Level

- *A specific instantiation of the test process*
- *Also defined as groups of test activities that are organised and managed together*
- For each test level following should be clearly identified
 - Specific test objectives for the test level
 - The test basis, the work products used to derive test conditions and test cases
 - Test Object
 - The typical defects and failures that we are looking for at this level
 - Specific approaches and responsibilities for this test level
 - Test Environment details

Component testing

- Is also called as module testing and unit testing
- ***A test level that focuses on individual hardware or software components***
- Objectives of component testing
 - Reducing risk by testing high risk components more extensively
 - Verifying whether the functional and non-functional behaviours of the component are as they should be
 - Building confidence in the quality of the component
 - Includes measurement of structural coverage of tests
 - Finding defects in component
 - Preventing defects from escaping to later stages
- Stubs and drivers are used if interfaces are not developed yet
- In agile development automated component regression tests are run frequently to give confidence that new additions or changes to component have not caused existing components or links to break

Component testing

- Test basis
 - Detailed design
 - Code
 - Data model
 - Component specification if available
- Test Objects
 - Component themselves, units or modules
 - Code and data structures
 - Classes
 - Database models

Component testing

- Typical defects and failures
 - Incorrect functionality
 - Data flow problems
 - Incorrect code or logic

Test Driven Development (TDD)

- Prepare and automate test cases before coding
- It's done in agile development
- First thing developer does is create automated tests for the component
- Just enough tests are written until those tests pass
- This may involve fixing defects found by test tests and refactoring the code
- This approach may also help build only what is needed rather than a lot of functionality which is not really wanted

Integration testing

- ***A test level that focuses on interactions between components or systems***
- ***Component Integration testing is The integration testing of components***
 - Usually done after component testing
 - Dependent on integration strategy like bottom-up, top-down, big-bang
 - In Agile it is usually part of CI/CD
- ***System Integration testing is the integration testing of systems***
 - Tests the interactions between different systems, packages and microservices and other external services

Integration testing

- Objectives of integration testing
 - Reducing risk by testing high risk integrations first
 - Verifying whether the functional and non-functional behaviours of the interfaces are as they should be
 - Building confidence in the quality of the interfaces
 - Finding defects in interfaces themselves or component or system being tested together
 - Preventing defects from escaping to later stages
- Test Basis
 - Software and system design
 - Sequence diagram
 - Interface and communication protocol specifications
 - Use cases
 - Architecture at component level or system level
 - Workflows
 - External interface definitions

Integration testing

- Test Objects
 - Sub systems
 - Databases
 - Infrastructure
 - Interfaces
 - APIs
 - Microservices
- Typical defects and failures of component integration testing
 - Incorrect data, missing data or incorrect data encoding
 - Incorrect sequencing or timing of interface calls
 - Interface mismatch
 - Failures in communication between components
 - Unhandled or improperly handled communication failures b/w components
 - Incorrect assumptions about the meaning, units or boundaries of the data being passed between components

Integration testing

- Typical defects and failures of System integration testing
 - Incorrect message structures between systems
 - Incorrect data, missing data or incorrect data encoding
 - Interface mismatch
 - Failures in communication between systems
 - Unhandled or improperly handled communication failures between systems
 - Incorrect assumptions about the meaning, units or boundaries of the data being passed between the systems
 - Failure to comply with mandatory security regulations

Integration testing

- Specific approaches and responsibilities
 - Big-bang approach
 - Incremental approach
 - Top-down
 - Bottom-up
 - Functional incremental
- Big-bang approach
 - All components and system are integrated simultaneously and everything is tested as a whole
 - Advantage is that everything is completed before integration no need to simulate any part
 - Disadvantage is that its is time consuming and difficult to trace the failures with late integration

Integration testing

- Incremental approach
 - All programs are integrated one by one
 - Tests are carried out after each step
 - Advantage is that defects are found early and relatively easy to find the cause
 - Disadvantage is that it's time consuming and stubs and drivers have to be developed
 - Top-down approach
 - Testing starts from top and works to the bottom following the control flow or architectural structure
 - Component or system are replaced by stubs
 - Bottom-up approach
 - Testing starts from the bottom of the control flow upwards. component s and system are replaced by drivers
 - Functional incremental
 - Integration testing takes place based the function or functionality as documented in functional spec

System Testing

- ***A test level that focuses on verifying that a system as a whole meets specified requirements***
- Test focus is on end to end tasks system should performance including non functional aspect such as performance
- According ISTQB definition system test is about verification, but it actual includes validation as well
- Test Environment preferably a mock of production environment as much as possible
- System test is carried out by test team or in some cases business analysts
- Objectives of system testing
 - Reducing risk
 - Verifying whether or not functional and non functional behaviours of the system are as they should be
 - Validating that the system is complete and will work as it should and as expected
 - Building confidence in quality of the system as a whole
 - Finding defects
 - Preventing defects from escaping to later stages or production

System Testing

- Test Basis
 - Software and system requirement specifications
 - Risk analysis reports
 - Use cases
 - Epics and user stories
 - Models of system behaviour
 - State diagrams
 - System and user manuals
- Test object
 - Applications
 - Hardware and software systems
 - Operating systems
 - System under test SUT
 - System configuration and configuration data

System testing

- Typical defects and failures
 - Incorrect calculations
 - Incorrect or unexpected system functional or non functional behaviour
 - Incorrect control and/or data flows within the system
 - Failure to properly and completely carry out end to end functional tasks
 - Failure of system to work properly under production environment
 - Failure of system to work as described in system and user manual

Acceptance testing

- ***A test level that focuses on determining whether to accept the system***
- Formal testing with respect to user needs, requirements and business processes conducted to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers or other authorised entity to determine whether or not to accept the system
- Objectives of acceptance testing
 - Establishing confidence in the quality of the system as a whole
 - Validating that the system is complete and will work as expected
 - Verifying that functional and non functional behaviours of the system are as specified

User Acceptance testing

- ***A type of acceptance testing performed to determine if intended users accept the system***
- Is done by end users
- The focus is on making sure that the system is really fit for purpose and ready to be used by real intended users of the system
- Is done on real environment or simulated operational environment
- The system needs to fulfill the requirements and meet their needs
- The user focuses on their business processes which they should be able to perform with a minimum of difficulty cost and risk

Operational acceptance testing

- ***A type of acceptance testing performed to determine if operations and/or systems administration staff can accept a system***
- Focuses on operations and may be performed by system admins
- The main purpose is to give confidence to the system admins or operators that they will be able to keep the system running and recover from adverse events quickly and without additional risk
- Performed in simulated production environment looking at aspects such as
 - Testing of backups and restoration of backups
 - Installing/uninstalling and upgrading
 - Disaster recovery
 - User management
 - Maintenance task
 - Data loading and migration tasks
 - Checking for security vulnerabilities
 - Performance and load testing

Contractual and regulatory acceptance testing

- ***Contractual acceptance testing is A type of acceptance testing performed to verify whether a system satisfies its contractual requirements***
 - Contractual testing is done by user or independent testers
 - Contractual testing is usually done when the company has developed software based on contract and acceptance criteria are already agreed upon
- ***Regulatory acceptance testing is A type of acceptance testing performed to determine the compliance of the test object***
 - Is carried out in order to verify if the system conforms to relevant laws, policies and regulations
 - Usually carried out along with a person from regulatory authority watching the test behaviour

Alpha and beta testing

- Typically used for COTS (Commercial off the shelf) software
- ***Alpha testing is a type of acceptance testing performed in the developer's test environment by roles outside the development organization***
- ***Beta testing is a type of acceptance testing performed at an external site to the developer's test environment by roles outside the development organization***
 - More visible, increasingly popular to be done remotely
 - Done under real world working conditions
 - Crowd testing where people or potential users from all over the world remotely test and app

Acceptance Testing

- Test Basis
 - Business process
 - User or business requirements
 - Regulations ,legal contracts and standards
 - Use cases
 - System requirements
 - System or user documentation
 - Installation procedure
 - Risk analysis reports

Acceptance testing

- Additional test basis for OAT (Operational acceptance testing)
 - Backup and restore recovery procedures
 - Disaster recovery procedures
 - Non functional requirements
 - Operation documentation
 - Deployment and installation instructions
 - Performance targets
 - Database packages
 - Security standard or regulations

Acceptance testing

- Test objects
 - System under Test SUT
 - System configuration and configuration data
 - Business process for full integrated systems
 - Recovery systems and hot sites
 - Operational and maintenance processes
 - Forms
 - Reports
 - Existing and converted production data

Acceptance testing

- Typical defects and failures
 - System workflow does not meet business or user requirement
 - Business rules are not implemented correctly
 - System does not satisfy contractual or regulatory requirements
 - Non functional failures such as security vulnerabilities, inadequate performance efficiency under high load, or improper operation on supported platform

Test Types

- Typical objective of Testing include
 - Evaluation of functional quality for example whether a function or feature is complete correct and appropriate
 - Evaluating a non functional quality characteristics for example reliability , performance, security, compatibility and usability
 - Evaluating whether the structure or architecture of the component or system is correct complete and as specified
 - Evaluating the effects of changes looking at both confirmation and regression
- ***A group of test activities based on specific test objectives aimed at specific characteristics of a component or system***
- Different characteristics would be
 - Functionality of component or system
 - Non functional aspect such as performance, reliability, security, usability, compatibility
 - Architecture
 - Related to changes

Functional testing

- ***Testing performed to evaluate if a component or system satisfies functional requirements***
- Functional testing is done based on functional or business requirements at different test levels
- There are two approaches
 - Requirement based testing
 - Uses specification of functional requirements for the system as basis for designing tests
 - Business process based testing
 - Uses knowledge of business process for the system as basis for designing tests
- Thoroughness of the functional testing can be measured by a covered measure based on elements of the function that we can list. Let's say we have 10 menu options then if at least one test covers each option we can say 100% coverage. However it depends on traceability.

Non Functional Testing

- ***Testing performed to evaluate that a component or system complies with non-functional requirements***
- Is performed at all levels
- Test non functional characteristics of the system like performance, load, stress, usability, maintainability, reliability, portability, security
- Measure of how well the system behaves
- Most black box testing techniques can be used for designing tests such as boundary value analysis, equivalence partitioning etc
- Thoroughness of non functional testing can be measured by the coverage of non functional elements

White box testing

- ***Testing based on an analysis of the internal structure of the component or system***
- Also called as clear-box testing, code-based testing, glass-box testing, logic-coverage testing, logic-driven testing, structural testing, structure-based testing
- White box testing is most often used as a way of measuring the thoroughness of testing through the coverage of a set of structural elements or coverage items such as system architecture, code, control flows, business process and data flows
- Usually done at component or component integration test level
- However it can be done at system or system integration test or acceptance test level with test basis as business model
- At component level there are readily available tools to measure code coverage

Change related testing

- Type of testing triggered by change in software in terms of bug fixes, functionality enhancements or change or requirements
- ***Confirmation testing is A type of change-related testing performed after fixing a defect to confirm that a failure caused by that defect does not reoccur***
 - We ensure to test the bug fix with exact steps, data and environment
- ***Regression testing is A type of change-related testing to detect whether defects have been introduced or uncovered in unchanged areas of the software***
 - Regression tests are done at all levels
 - Regression are usually best candidate for automation
 - Every organization usually has its own regression suite and it's better to update the suite periodically based on changes enhancements or deletion of part of software

Maintenance Testing

- ***Testing the changes to an operational system or the impact of a changed environment to an operational system***
- Triggers
 - Modification
 - Include planned enhancements, corrective and emergency changes, change of environment such as planned operating system or database upgrade or patches to newly exposed or discovered vulnerability of OS and upgrade to CTOS, modification to hardware or devices
 - Migration
 - Migration to new environment as well as changed software
 - Retirement
 - Data migration or archiving
- Factor affecting the scope of maintenance testing
 - Degree of risk of the change for ex self contained change is lower risk than a part of system which communicates with other
 - Size and complexity of existing system
 - Size of change

Impact analysis and regression testing

- ***Impact analysis is identification of all work products affected by a change, including an estimate of the resources needed to accomplish the change***
- Maintenance test usually consists of two steps
 - Testing the change
 - Testing the regression
- Advantages of impact analysis
 - Understand how much regression needs to be done
 - Decide if the change has to be done or not

Impact analysis

- Factor making impact analysis difficult
 - Specification are out of date or missing
 - Test cases are not documented or missing
 - Bi-directional traceability between tests and test basis has not been maintained
 - Tool support is weak or non existent
 - The people involved do not have domain or system knowledge
 - The maintainability has not been taken into enough consideration during the development

Static Testing

Static analysis and static testing

- *Static Analysis is the process of evaluating a component or system without executing it, based on its form, structure, content, or documentation*
- *Static testing is testing that does not involve the execution of a test item*

Work products that can be examined by static testing

- Any type of specification - Business, functional and security requirements
- Epics, user stories, acceptance criteria
- Code
- Testware
- User guide, help text, wizards
- Web Page (ex links are broken or not)
- Contract, project plans, schedule and budgets
- Models such as activity diagram or other models used in Model based testing

Static analysis

- Static analysis of software code is done using a tool to analyze the code with respect to acceptance criteria
 - Static analysis tools can find dead code
 - Static analysis tool can find variables which are used before initializing which can be hard to find using dynamic testing
 - Static analysis tools can analyze natural text in requirements find typos and readability

Benefits of static testing

- Since static testing can start early in life cycle, early feedback on quality issues can be established.
- By detecting defects at an early stage, rework costs are most often relatively low.
- Defects are more efficiently detected and corrected
- Defects that are not easily found by dynamic testing
- Defect in future design and code are prevented by uncovering inconsistencies, ambiguities, contradictions, omissions, inaccuracies and redundancies in requirements
- Since rework effort is substantially reduced, development productivity likely to increase
- Reduced development cost
- Reduced testing cost and time
- Reduced cost of quality over software's lifetime
- Improved communication within the team

Difference between static and dynamic testing

- In Static testing code is not executed
- As dynamic testing can be done only when the software code is ready, there are lot of work products which are not covered by dynamic testing
- Static testing finds only defects
- Static testing is more informative and educational and improved team communication and collaboration
- Code is executed
- Dynamic testing can only be done once software code is ready
- Dynamic testing finds failures and then when we analyse we can call it a defect

Review

- *Review is a type of static testing in which a work product or process is evaluated by one or more individuals to detect defects or to provide improvements*
- *Information Review is a type of review that does not follow a defined process and has no formally documented output*
- *Formation Review is a type of review that follows a defined process with a formally documented output*

Benefits of early and frequent stakeholder feedback

- Frequent feedback is vital for Agile development teams to understand whether the team is going in the direction as expected. The product increment developed by Agile team will be subjected to stakeholder review and any feedback may be appended to the product backlog, which can be prioritized at the discretion of the Product Owner.
- One of the best ways to provide frequent feedback is through continuous integration
 - During Waterfall development, each activity like design, development, testing is considered as a phase, but in agile all the activities is done in small chunks every iteration.
 - In waterfall, the customer can see the software working at the end of the project, and at that stage the changes are very costly and involve significant rework.
 - Instead, if the feedback is obtained at the end of every iteration, it may be very easy for the team to make up the feedback and go along.
 - It also serves as an important tool to adopt modifications in the application.
 - Highest business value features are delivered first to the customer by development teams through the use of early and frequent feedback.
- It also in a way helps the team to measure its own capacity so that they do not grossly over commit thus building high degree of transparency into the planning process

Benefits of early and frequent stakeholder feedback

- Breaking all the requirements in small pieces of individual work items will avoid mistakes that may be too costly to fix later
- Customer availability for any questions to the team makes the product development robust, so that team exactly build what customer wants
- The team will be able to deliver at a constant and sustainable pace
- Sharing agile productivity metrics helps the team to understand the gaps better so that they can find ways to improve themselves

Review Process

- Planning
- Review initiation
- Individual review
- Issue communication and analysis
- Fixing and reporting

Planning

- People involved (Review leader and Author)
- Define the scope of the review
- Estimating effort and timeframe for the review
- Identifying review characteristics such as the type of review with roles activities and checklist
- Selecting people to participate in the review and allocating roles to reviewer
- Defining entry and exit criteria for formal review types
- Checking that entry criteria are met before the review starts

Initiate Review

- People involved (Facilitator, author)
- Distributing the work products and any other relevant materials such as logging forms, checklists or related work products
- Explaining the scope, objectives, process, roles and work products to be the participants
- Answering any questions that participants may have about the review

Individual review

- People involved (Reviewers)
- Reviewing all or part of work documents
- Noting potential defects, recommendations and questions

Issue communication and analysis

- Communicating identified potential defects for example in a review meeting
- Analysing potential defects, assigning ownership and status to them
- Evaluating and documenting quality characteristics
- Evaluating the review findings against the exit criteria to make review decision

Fixing and Reporting

- Creating defect reports for those findings that require changes
- Fixing defects found in the work product reviewed
- Communicating defects to the appropriate person or team
- Recording updated status of defects, potentially including the agreement of the comment originator
- Gathering metrics, for ex defects fixed, deferred
- Checking that exit criteria are met
- Accepting the work product when the exit criteria are reached

Possible entry criteria

- A short check of a work product sample by review leader does not reveal major defects
- The work product to be reviewed is available with line numbers
- The work product has been cleaned up by running any applicable automated checks, such as static analysis or spelling and grammar assessments
- References needed for the review are stable and available
- The work product author is prepared to join the review team and feels confident with quality of the product

Review meeting

- Logging in review meeting
 - All the defects are first logged by author or scribe with the severity
 - No discussion is done during this part, if discussion is needed it is noted
 - Spelling mistakes are not logged they are communicated later to author
- Discussion part of review meeting
 - Discussion of the noted defects
- Decision making part of review meeting
 - Decision and action items are taken down
 - Decision for repeating review can also be taken based on format exit criteria

Roles and responsibilities

- Author
 - Creating work product under review
 - Fixing defects in the work product
- Manager
 - Ensuring that reviews are planned
 - Deciding on the execution of reviews
 - Assigning staff, budget and time
 - Monitoring ongoing cost effectiveness
 - Executing control decisions in the event of inadequate outcomes

Roles and responsibilities

- Facilitator (Moderator)
 - Ensuring the effective running of review meetings
 - Mediating if necessary between various points of view
 - Being the person upon whom the success of review often depends
- Review Leader
 - Taking overall responsibility of the review
 - Deciding who will be involved
 - Focuses mainly on review happening and organise the people involved and may not be part of review

Roles and responsibilities

- Reviewers
 - Being a subject matter expert
 - Identifying potential defects in the work product under review
 - Representing different perspectives as requested for ex tester, dev, user, operator, business analyst
- Scribe
 - Collating potential defects found during the individual review activity
 - Recording new potential defects, open points and decisions from the review meeting

Types of reviews

- Informal reviews
- Walkthrough
- Formal review
- inspection

Informal Review

- Main purpose/Objective : Detecting potential defects
- Possible additional purpose : Generating new ideas or solutions, quickly solving minor problems
- Not based on a format review process
- May not involve review meeting
- May be performed by colleague of the author
- Results may be documented
- Varies in usefulness depending on the reviewer
- Use of checklist is optional
- Very commonly used in agile development

Walkthrough

- ***A type of review in which an author leads members of the review through a work product and the members ask questions and make comments about possible issues***
- Main purpose - Find defects, improve the software product, consider alternative implementations, evaluate conformance to standards and specifications
- Possible additional purposes : Exchanging ideas about techniques or style variations, training of participants, achieving consensus
- Individual preparation before the meeting is optional
- Review meeting is typically led by author of work product
- Use of scribe is mandatory
- Use of checklists is optional
- May take the form of scenarios, dry runs or simulations
- Potential defect logs and review reports may be produced
- May vary in practice from quite informal to formal

Technical review

- ***A formal review by technical experts that examine the quality of a work product and identify discrepancies from specifications and standards***
- Main purpose : Gaining consensus , detecting potential defects
- Possible further purpose: Evaluating quality and building confidence in the work product, Generating new ideas, motivating and enabling authors to improve future products, considering alternative implementation
- Reviewers should be technical peers, technical experts in the relevant discipline
- Individual preparation before the meeting is required
- Review meeting is optional
- Review meeting is ideally led by a trained facilitator
- Scribe is mandatory and ideally not the author
- Use of checklist is optional
- Potential defect logs and review reports are typically produced

Inspection

- ***A type of formal review that uses defined team roles and measurement to identify defects in a work product, and improve the review process and the software development process***
 - Main purpose : Detecting potential defects, evaluating quality and building confidence in work product, preventing future defects through author learning and root cause analysis
 - Possible further purpose : Motivating and enabling authors to improve future work products and software development process, achieving consensus
 - A defined process is followed, with formal documented outputs, based on rules and checklists
 - There are clearly defined roles
 - Individual preparation before review meeting is required
 - Reviewers are either peers of author or expert in other disciplines that are relevant to the work product
 - Specified entry and exit criteria are used
 - A scribe is mandatory
 - Review meeting is led by trained facilitator
 - Author cannot review leader, facilitator, reader or scribe
 - Potential defect logs and review reports are produced
 - Metrics are collected and used to improve the entire software development process, including inspection process

Review Techniques

- Ad hoc reviewing
- Checklist based reviewing
- Scenario based reviewing
- Role based reviewing
- Perspective based reviewing

Ad hoc reviewing

- ***A review technique performed informally without a structured process***
- Reviewer looks at work product and note possible defects
- Reviews are carried out based on experience and attention to detail

Checklist based reviewing

- ***A review technique guided by a list of questions or required attributes***
 - List of questions to look for is made as a checklist
 - Checklist should be small and high risks items at first
 - Review checklist regularly to update
 - Don't just stick to checklist, take inspiration and look outside as well

Scenario based reviewing

- ***A review technique in which a work product is evaluated to determine its ability to address specific scenarios***
- We look at the way the system is expected to be used, which is an important aspect of validation
- Unlike checklist where we verify individual aspect, scenario based review focuses on how system will be used from specific perspective
- Don't just look at given scenarios or use cases provided, also apply knowledge to explore more

Role based reviewing

- ***A review technique in which a work product is evaluated from the perspective of different stakeholders***
- Similarly like scenario based, Work product is looked at from different personas like different users who can use the system, an elderly person, unmarried who like skiing
- Roles can also be organisational like admin, tester, end user etc
- Sometime review can be distributed like one person review one user story

Perspective based reviewing

- **A review technique in which a work product is evaluated from the perspective of different stakeholders with the purpose to derive other work products**
- Similar to role based, but rather than playing specific role, reviewer typically tries to perform the tasks on a high level that they would be doing with the work product under review.
- Another approach is to use work product to generate other work products from it. While doing it we will uncover the gaps and mistakes

Organisational success factors for reviews

- Have a champion person who will lead the process on project or organisational level
- Have clear objective
- Pick the right review type and technique
- Review materials need to be kept up to date
- Limit the scope of review
- It takes time
- Management support

People related success factors for review

- Pick the right reviewers
- Use testers
- Each participant does his work well
- Limit the scope of review and pick the things that really count
- Defect should be welcomed
- Review meetings are well managed
- Trust is critical
- How you communicate is important
- Follow the rules but keep it simple
- Train participants
- Continuously improve the process and tools
- Just do It!

Test techniques

Overview

- ***Test technique is A procedure used to define test conditions, design test cases, and specify test data***
- The purpose of test technique is to identify test conditions , test cases and test data

Factor affecting the selection of test technique

- Type of component or system. Ex input field - Equivalence partition or BVA
- Component or system complexity
- Regulatory standards
- Customer or contractual requirement
- Test Objectives
- Available documentation
- Tester knowledge and skills
- Available tools
- Time and budget
- SDLC Model
- Expected use of software
- Previous experience with using test technique
- Types of defect expected from the component under test

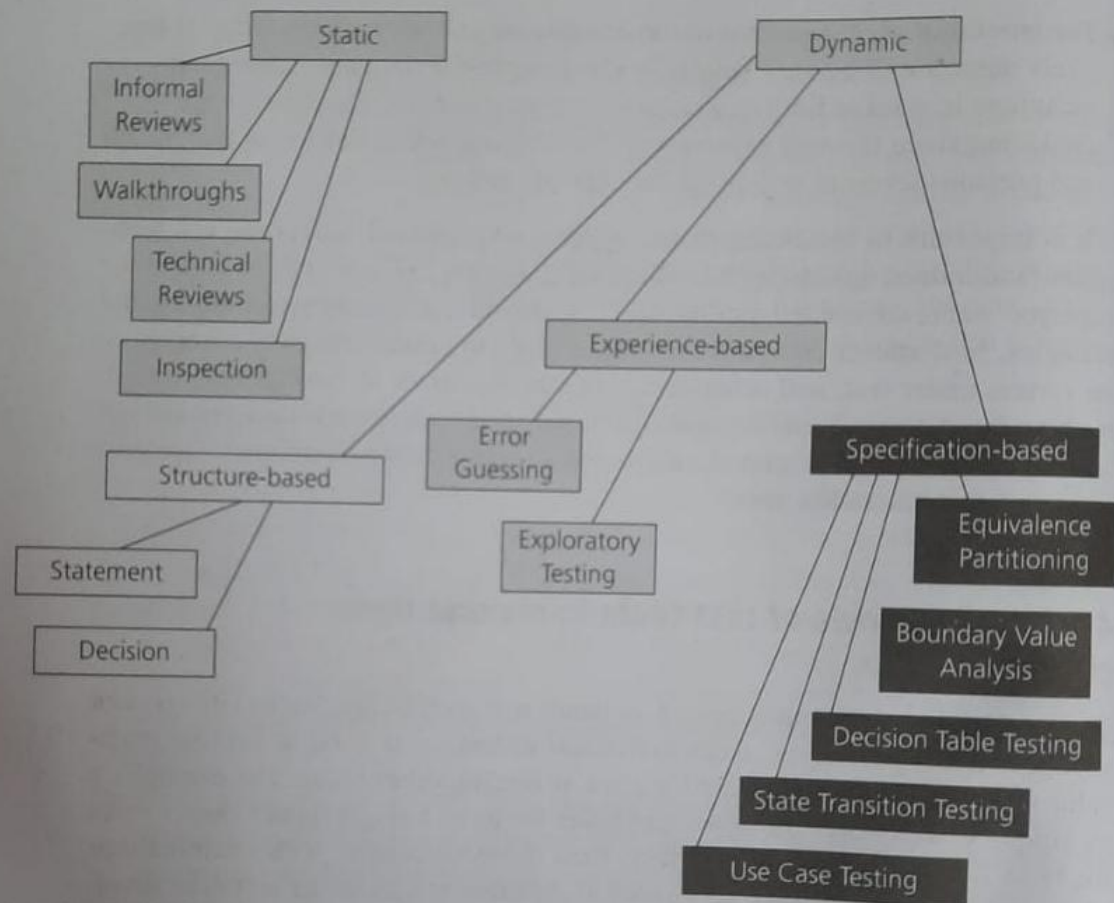


FIGURE 4.1 Test techniques

Black box test techniques

- ***A test technique based on an analysis of the specification of a component or system without reference to its internal structure***
- Common characteristics of black box test
 - Test Condition, test cases and test data are derived from test basis such as Software requirement, specifications, use cases and user stories
 - Test cases may be used to detect gaps between requirement and corresponding implementation as well as the deviation from those requirements
 - Coverage is measured based on the items tested in the test basis and test technique applied to the item

White box test technique

- ***A test technique only based on the internal structure of a component or system***
- Common characteristics of white box
 - Test conditions, test cases and test data are derived from test basis that may include, code, software architecture, detailed design or any other source of information regarding the structure of the software
 - Coverage is measured based on the items tested within a selected structure, for example the code statements, the decisions, the interfaces.
 - White box test techniques determine the path through the software that either was taken or that you want to be taken, and this is determined by specific inputs to the software.

Experience based test technique

- ***A test technique based on the tester's experience, knowledge and intuition***
- People's knowledge, skills and background are prime contributor to the test conditions, test case and test data
- People's knowledge both on system and type of probable defects
- Can be used when specification is not there or inadequate or high time pressure
- Should not be used for high risk applications

Black box test techniques

- Equivalence partitioning
- Boundary Value analysis
- Decision table testing
- State Transition testing

Equivalence partitioning

- ***A black-box test technique in which test conditions are equivalence partitions exercised by one representative member of each partition***
- Idea behind the technique is to divide a set of test conditions into groups or sets where all elements of the set can be considered the same, so the system should handle them equivalently, hence Equivalence partitioning

Boundary value analysis

- ***A black-box test technique in which test cases are designed based on boundary value***
- Two value boundary
 - One invalid EP and one Boundary value
- Three value boundary
 - One invalid EP, one boundary value(which is also valid EP) , next value

Decision table testing

- ***A black-box test technique in which test cases are designed to exercise the combinations of conditions and the resulting actions shown in a decision table***
- Is a Good way to deal with combinations of things
- Using Decision table for Test design
 - First task is to identify suitable function or subsystem that has a behaviour which reacts according to a combination of inputs or events
 - Once you have identified the aspects that need to be combined, then you put them into a table, listing all the combinations of True and False for each of the aspects
 - With 2 conditions, we will have Four combinations, with 3 its 8 and with 4 its 16 Combination
 - Then we Decide which ones can be eliminated and then write out test cases
 - Prioritize test cases and execute them

State Transition testing

- ***A black-box test technique in which test cases are designed to exercise elements of a state transition model***
- Is used where some aspect of the system can be described in what is called a finite state machine
- Any system where you get different output for same input depending on what has happened before is finite state system

Use case testing

- ***A black-box test technique in which test cases are designed to exercise use case behaviors***
- Mostly used for system or acceptance testing
- Coverage is measured by % of use case behaviours tested by total number of use case behaviours

White box test technique

- Statement testing
- Branch (Decision) Testing

Coverage

- ***The degree to which specified coverage items are exercised by a test suite, expressed as a percentage***
- In General terms coverage is misunderstood we use it for asking how much testing is completed, that can be called as **test completeness** not **coverage**
- For developers coverage means, Statement or branch coverages

Coverage in Black box testing

- Coverage in black box testing would apply at whichever test level it used
- Equivalence partitioning - percentage of EPs exercised
- Boundary value analysis - percentage of BVs exercised
- Decision table testing - percentage of business rules or decision table columns tested
- State transition testing
 - Percentage of states visited
 - Percentage of valid transitions exercised
 - Percentage of pairs of valid transition exercised
 - Percentage of invalid transitions exercised

How to measure coverage - White box testing

- Decide on structural element to be used , that is , the coverage items to be counted
- Count the structural elements or items
- Instrument the code
- Run the tests for which coverage measurement is required
- Using the output from instrumentation, determine the percentage of elements or items exercised

Statement testing and coverage

- ***A white-box test technique in which test cases are designed to execute statements***

Statement Coverage = (Number of statements exercised / Total number of statements) x 100

Decision testing and coverage

- ***A white-box test technique in which test cases are designed to execute decision outcomes***

Decision testing coverage = (Number of decision outcomes exercised / Total number of decision outcomes) X 100

- 100% decision coverage always guarantees 100% statement coverage not vice versa

Value of White box testing

- The value of statement and decision testing is in seeing what new tests are needed in order to achieve higher level of coverage, whatever dimension of coverage we are looking at
- Coverage is a partial measure of some aspect of thoroughness of testing

Experience based testing

- Error guessing
- Exploratory testing
- Check list based testing

Error Guessing

- ***A test technique in which tests are derived on the basis of the tester's knowledge of past failures, or general knowledge of failure modes***
- Error guessing may be based on
 - How the application has worked in the past
 - What types of mistakes the developer tend to make
 - Failures that have occurred in other applications
- Formal approach is to list possible defects or failure and to design tests that produce them
- Possible things to test Division by zero, blank or no input, empty files and wrong kind of data

Exploratory testing

- ***An approach to testing in which the testers dynamically design and execute tests based on their knowledge, exploration of the test item and the results of previous tests***
- Planning involves creation of test charter, a short declaration of the scope of short time boxed test effort , the objectives and possible approaches
- Test design and execution activities are done in parallel
- Less documentation of design, however test cases are written during execution
- This is also called as Session based testing as one of the way to organise and manage testing is by conducting session
- This employs learning based approach to test. The tester constantly making decision on what to test next based on the system behaviour

Check list based testing

- ***An experience-based test technique in which test cases are designed to exercise the items of a checklist***
- As this is experience based, the experience is summarized and documented in checklist
- The checklist may be based on
 - Experience of the tester
 - Knowledge for example what is important for the user
 - Understanding of why and how software fails
- Tester can also modify the checklist while testing
- As the documentation is at high level, different testers can test different scenarios based on their understanding and creativity. That's the unique point of checklist based testing

Collaboration based test technique

- ***An approach to testing that focuses on defect avoidance by collaborating among stakeholders***
- This is mainly focuses on defect avoidance by collaboration and communication

Collaborative User Story writing

- Card

- User story written like a card

As a blogger, I want to show a Twitter share button at the end of each blog page so that my readers can share this article in their Twitter feed with one click if they want to

- Conversation

- User story doesn't contain all information, We arrange this conversation during Sprint Planning meeting
 - At that conversation must address all questions like- “what”, “why”, “when”, “who”, “how”
 - Also at this session they will discuss about every details that are required to implementing the solution for the story

- Confirmation

- After confirmation we will have a solid understanding what exactly we have to do for successfully completing the story
 - We organise work step by step and achieve them. It is accepted to be complete when acceptance criteria is met 100% and user story is accepted to be complete when acceptance tests are 100%

<https://medium.com/@mojamcpds/card-conversation-confirmation-d88c074dd58c>

Acceptance Criteria

- ***The criteria that a component or system must satisfy in order to be accepted by a user, customer, or other authorized entity***
- They are used to:
 - Define the scope of the user story
 - Reach consensus among the stakeholders
 - Describe both positive and negative scenarios
 - Serve as basis for user story acceptance testing
 - Allow accurate planning and estimation

Scenario oriented acceptance criteria

- Scenario-oriented acceptance criteria are a way of defining the expected behavior of a user story by describing specific scenarios or situations in which the functionality of the story should work.
 - Understand user story
 - Identify key scenarios
 - Use Given , When , then format
 - Be specific and concrete
 - Cover positive and negative scenarios
 - Prioritize criteria
 - Consider Edge cases
 - Review with Stakeholders
- By writing scenario-oriented acceptance criteria, you provide a clear and comprehensive set of requirements that can guide the development and testing of the user story. This approach also helps in preventing misunderstandings and ensures the delivered functionality meets the intended user needs

Rule Oriented Acceptance criteria

- Rule-oriented acceptance criteria are a way of defining the conditions and constraints that must be satisfied for a user story to be considered complete. Unlike scenario-oriented acceptance criteria, which focus on specific scenarios or situations, rule-oriented criteria concentrate on setting rules and boundaries for the functionality being developed
 - Understand User story
 - Identify the rules
 - Make rules specific and measurable
 - Use clear language
 - Prioritize and group rules
 - Consider negative scenarios
 - Collaborate with stakeholders
 - Review with the team

Acceptance test driven development

- ***A collaboration-based test-first approach that defines acceptance tests in the stakeholders' domain language***
 - Whole development team and business stakeholders are involved
 - Firstly it is discussed what is required by user, then the user stories are created with acceptance criteria
 - Developer first writes test for those criteria
 - Then developer writes code to make those test pass
 - If tests fail, developer refactoring his code to make those tests pass

Deriving test case based on ATDD

- Define acceptance criteria
- Derive test case from acceptance criteria
- Write test cases and automate if possible
- Implement features
- Run the tests
- Verify results

Managing test activities

Test Organisation - independence in testing

- First level - Developer testing own code
- Second level - Developer or testing peer testing code reporting to same manager
- Third level - Separate testing team within development phase
- Fourth level - Separate test team outside development team or organisation

Advantages of independence in testing

- Testers mindset of critical thinking and professional pessimism
- With Independent structure for test organisation, testers can honestly point at what's wrong
- With independent team budget is separate hence can decide on training, tools and other things

Potential drawbacks of independence

- There is a gap between tester and test team, tester and developer giving rise communication problem
- Some times test team is not ready to align with business priority
- Some stakeholders might think independent test team as a hindrance to progress
- Developer might not test their code as there is test team
- Tester might face information clarity issue and it will drag project
- Testers might miss some bugs due to lack of information of design or code etc

Test manager tasks

- ***The person responsible for project management of testing activities, resources, and evaluation of a test object***
- The individual who directs, control, administers, plans, regulates, the evaluation of test object
-

Testers task

- *A person who performs testing*

-

Skills of tester

- Application or business domain knowledge
- Knowledge of technology being used
- Testing knowledge

Purpose and content of test plan

- ***Test Plan is a Documentation describing the test objectives to be achieved and the means and the schedule for achieving them, organized to coordinate testing activities***
- Purpose
 - Writing a test plan guides our thinking
 - Test planning process or the plan itself serve as vehicles for communicating with other members of the project team, testers, peers, managers and other stakeholders
 - Test plan helps us manage change
 - Whenever there is change we can try and update the plan
- We can also master test plan to tackle different test levels and test types

What is in the test plan

- What is in the scope and what is out of the scope for this testing effort
- What are the test objectives
- What are the important project and product risks
- What is overall approach of testing in this project
- How will test activities be integrated and coordinated into the software life cycle activities
- How do we decide what to test what people and other resources are needed to perform test activities and how test activities will be carried out
- What constraints affect testing (Budget, timeline)
- What is most critical for this project and product
- Which aspect of product are more testable
- What should be the overall test execution schedule, how should we decide the order in which to do test analysis, design, implementation, execution and evaluation of specific tests, either on specific dates or in the context of an iteration
- What metrics will be used for test monitoring and control, how will they be gathered and analysed
- What is the budget for all testing activities
- What should be level of details and structure of test documentation

What's in the test plan

- What operations support required for test environment
- What kind of information must be delivered to the maintenance team at the end of testing
- How precisely should the testers write test design, procedure and cases. How much should they leave to the judgement of the tester during test execution, and what are the reproducibility issues associated with this decision
- What kind of templates can testers use for the various documents they will produce. How do those documents relate to one another

Test Strategy and test approach

- ***Test Policy is A high-level document describing the principles, approach and major objectives of the organization regarding testing***
- ***Test Strategy is A description of how to perform testing to reach test objectives under given circumstances***
- ***Test approach is The manner of implementing testing tasks***
- Factors to consider while working test approach
 - Testing objectives
 - Project goals
 - Overall risk management
 - Project, product and organization context, issue related to risks, hazards and safety, available resources, teams level of skills, the technology involved, nature of system under test and applicable regulations

Test Strategies

- Analytical
- Model based
- Methodical
- Process or standard compliant
- Directed
- Regression averse
- Reactive

Factors to consider while selecting best test strategy

- Risks
- Skills
- Objectives
- Regulations
- Product
- Business

Tester's contribution to release testing

- Release planning defines and redefines product backlog and may involve breaking down larger user stories into collection of smaller stories
- Release planning provides basis for test approach spanning all iterations and Release planning is high level
- Business representative, Product owners establish and prioritize user stories in collaboration with the team
- Testers are involved in
 - Defining testable user stories including acceptance criteria
 - Participating in project and quality risk analysis
 - Estimating testing effort associated with user story
 - Defining necessary test levels
 - Planning the testing for the release

Tester's contribution to iteration testing

- The team selection user story from prioritized release backlog, elaborates user story, performs risk analysis and estimates the work needed each user story
- Business representative or product owner should answer questions about each user story so that they can understand what to develop and test
- Testers are involved in
 - Detailed risk analysis of user story
 - Determining testability of the user stories
 - Creating acceptance tests for the user story
 - Breaking down user stories into tasks and subtasks
 - Estimating testing effort for all tasks
 - Identifying functional and non functional aspects of system to be tested

Issues with Release and iteration planning

- Release plans may change as the project needs, including individual user stories from the product backlog
- These changes can be challenging for testers. They must have enough test basis and test oracle for in each iteration for test development process
- Release and iteration planning should address, planning for both dev and test activities
- Issues to be addressed
 - Scope of testing, extent of testing for those areas on scope, the test goals and reasons for these directions
 - Team members who will carry out these activities
 - Test environment and test data needed
 - Timing, sequence, dependency and prerequisites for functional and nonfunctional test activities
 - The project and quality risks to be address

Entry Criteria (Definition of Ready)

- ***The set of conditions for officially starting a defined task***
- Entry criteria include the following
 - Availability of test basis
 - Availability of test items that have met the exit criteria for any previous test levels
 - Availability of test environment
 - Availability of necessary tools and other materials needed
 - Availability of test data and other necessary resources
 - Availability of staff for testing tasks
 - Availability of test object (Component to be tested)

Exit criteria (Definition of Done)

- ***The set of conditions for officially completing a defined task***
- Criteria
 - Tests : Planned, prepared, run, passed, failed blocked and deferred are acceptable
 - Coverage - The extent to which test basis, risk, functionality, supported configuration and software code have been tested
 - Defects
 - Quality
 - Money
 - Schedule
 - Risk

Test Execution schedule

- ***A schedule for the execution of test suites within a test cycle***
- Factors to consider while deriving test execution schedule
 - Priority of tests
 - Technical and logical dependencies of test

Test Estimation

- ***An approximation related to various aspects of testing***
 - For ex. Effort spent, completion date, costs involved, number of tests etc...
 - There are two ways to go about
 - Forward approach
 - Start with test planning and then subsequent activities
 - Write down tasks and subtasks in each activity and plan for them
 - Then move on to next activity
 - Backward
 - Start from the bottom
 - Let's say you want to performance testing. Then start from what all is needed for performance testing
 - May be test design, test implementation, test environment set up , test automation script creation
 - For each activity do the estimation

Factors influencing test effort

- Product characteristics
 - The risks associated with the product
 - The quality of test basis
 - The size of the product
 - The requirements for the quality characteristics such as usability, reliability, security etc
 - Test Complexity of product domain
 - The geographical distribution of the team
 - The required level of detail for test documentation
 - Requirements for legal and regulatory compliance

Factors influencing test effort

- Development process characteristics
 - Stability and maturity of the organization
 - The SDLC model used
 - The test approach
 - The tools used
 - Test test process
 - Time pressure

Factors affecting test effort

- People characteristics
 - Skills and experience of people
 - Team cohesion and leadership
- Test Results
 - Number and severity of defect found
 - Rework required

Test Estimation techniques

- Expert - based
 - Consulting the people who do the work
- Metrics based
 - Analyzing metrics from past project and industry data

Expert based technique

- Involves experience staff members to develop work-breakdown structure
- Understand each task, effort , duration, dependency and resource requirement
- This is also called bottom up approach
- Ex. planning poker approach

Metrics based approach

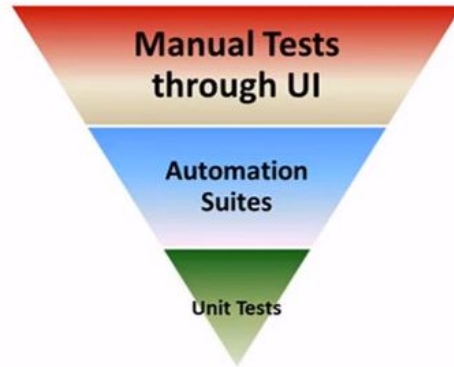
- Involves analyzing data from previous projects or industry data
 - Effort required for test case in previous data
 - Number of defects and time to remove them
 - In Agile based on velocity, how much do we take to complete a user story
 - Ex. Burndown chart technique, Defect removal model

Test case prioritization

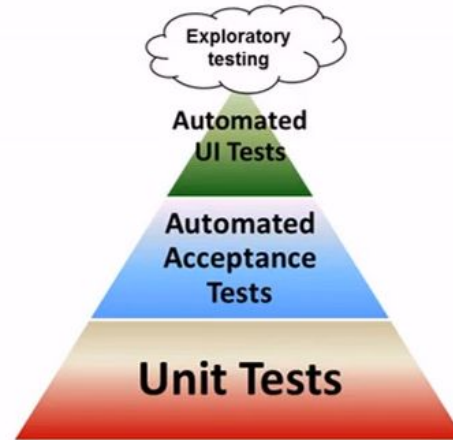
Test Pyramid

- ▶ A software system may be tested at different levels.
- ▶ Typical test levels are, from the base of the pyramid to the top, unit, integration, system, and acceptance.
- ▶ The test pyramid emphasizes having a large number of tests at the lower levels (bottom of the pyramid) and, as development moves to the upper levels, the number of tests decreases (top of the pyramid).
- ▶ Usually unit and integration level tests are automated and are created using API-based tools.
- ▶ At the system and acceptance levels, the automated tests are created using GUI-based tools.
- ▶ The test pyramid concept is based on the testing principle of early QA and testing (i.e., eliminating defects as early as possible in the lifecycle).

Test Pyramid



Traditional
(find bugs)




Agile
(prevent bugs)

Testing Quadrant

- ▶ The testing quadrants model, and its variants, helps to ensure that all important test types and test levels are included in the development lifecycle.
- ▶ This model also provides a way to differentiate and describe the types of tests to all stakeholders, including developers, testers, and business representatives.
- ▶ In the testing quadrants, tests can be business (user) or technology (developer) facing. Some tests support the work done by the Agile team and confirm software behavior.
- ▶ Other tests can verify the product. Tests can be fully manual, fully automated, a combination of manual and automated, or manual but supported by tools.

Testing Quadrants

The four quadrants are as follows:

- ▶ **Quadrant Q1** is **unit level**, **technology facing**, and supports the developers. This quadrant contains unit tests. These tests should be automated and included in the continuous integration process.
 - ▶ **Quadrant Q2** is **system level**, **business facing**, and confirms product behavior. This quadrant contains functional tests, examples, story tests, user experience prototypes, and simulations. These tests check the acceptance criteria and can be manual or automated. They are often created during the user story development and thus improve the quality of the stories. They are useful when creating automated regression test suites.
 - ▶ **Quadrant Q3** is **system or user acceptance level**, **business facing**, and contains tests that critique the product, using realistic scenarios and data. This quadrant contains exploratory testing, scenarios, process flows, usability testing, user acceptance testing, alpha testing, and beta testing. These tests are often manual and are user-oriented.
 - ▶ **Quadrant Q4** is **system or operational acceptance level**, **technology facing**, and contains tests that critique the product. This quadrant contains performance, load, stress, and scalability tests, security tests, maintainability, memory management, compatibility and interoperability, data migration, infrastructure, and recovery testing. These tests are often automated.
- 

Test Monitoring and control

- *Test monitoring is The activity that checks the status of testing activities, identifies any variances from planned or expected, and reports status to stakeholders*
- *Test control is The activity that develops and applies corrective actions to get a test project on track when it deviates from what was planned*

Metrics used in testing

- Purpose of test metrics
 - Give the test team and test manager feedback on how the testing work is going allowing opportunities to guide and improve testing and project
 - Provide project team with visibility about the test results and quality of test object
 - Measure the status of the testing, test coverage and test items against exit criteria to determine whether the test work is done and to assess effectiveness of test activities with respect to objectives
 - Gather data for use in estimation of future test efforts including adequacies of test approach

System Test Case Summary												
Cycle One												
Test ID	Test Suite/Case	Status	System Config	Bug ID	Bug RPN	Run By	Plan Date	Act Date	Plan Effort	Actual Effort	Test Duration	Test Comment
1.000	Functionality											
1.001	File	Fail	A	701	1	LTW	1/8	1/8	4	6	6	
1.002	Edit	Fail	A	709	1	LTW	1/9	1/10	4	8	8	
				710	5							
				718	3							
				722	4							
1.003	Font	Pass	B			IHB	1/10	1/10	4	4	4	
1.004	Tables	Warn	B	708	15	IHB	1/8	1/9	4	5	5	
1.005	Printing	Skip					1/10		4			Out of runway
	Suite Summary						1/10	1/10	20	23	23	
2.000	Performance/Stress											
2.001	Solaris Server	Warn	A,B,C	701	1	EM	1/10	1/13	4	8	24	Replan 1/11
2.002	NT Server	Fail	A,B,C	724	2	EM	1/11	1/14	4	4	24	Replan 1/12
				713	2							
				725	1							
2.003	Linux Server	Skip					1/12		4			Out of runway
	Suite Summary						1/12	1/14	12	12	48	
3.000	Error Handling/Recovery											
3.001	Corrupt File	Fail	A	701	1	LTW	1/8	1/9	4	8	8	
				706	2							
				707	4							
				709	1							
				710	5							
				713	2							
				712	6	LTW	1/9	1/10	4	6	6	
3.002	Server Crash	Fail	A	713	2							
				717	1							
	Suite Summary						1/9	1/10	8	14	14	
4.000	Localization											
4.001	Spanish	Skip										
4.002	French	Skip										
4.003	Japanese	Skip										
4.004	Chinese	Skip					1/0	1/0	0	0	0	
	Suite Summary											

FIGURE 5.1 Test case summary worksheet

such a worksheet would

Common metrics

- % of planned work done in test case preparation
- % of planned work done is test environment preparation
- Metrics related to test execution, such as the number of test cases run and not run, the number of test cases passed or failed, or number of test conditions passed or failed
- Metrics relating to defects, such as defect density , defects found and fixed, failure rate and confirmation test results
- The extent of test coverage achieved, measure against requirements, user stories, acceptance criteria, risks, code, configurations or other areas of interest
- Status of testing compared to various milestones, that is, task completion, resource allocation, usage and effort
- The economics of testing, such as costs and benefits of continuing test execution in terms of finding the next defect or running the next test

Purpose of test report

- To provide the status of testing activities to stakeholders
- Reporting any risk items
- Stating the quality of the testing

Content of test progress report

- The current status of the test activities and progress against the plan
- Factors that are currently impeding the progress of the testing
- Testing planned for next period to be including in the next test progress report
- Quality of test object as currently assessed by the testing

Content of test summary report

- A summary of testing performed
- Information about what occurred during the period the report covers
- Deviation from plan, with regard to schedule duration or effort of test activities
- Status of testing and product quality with respect to the exit criteria (DoD)
- The factors that have blocked or continue to block progress
- Relevant metrics, such as those relating to test cases, defects, test coverage, test activity progress and resource consumption
- Residual risks
- Reusable test products

Audience of test report

- Test Summary report sent to CEO or other high level management should be short and concise. With overview summary of main points. It may contain summary of defects of high and other priority levels and percentage of tests passed but information about budget and schedule would be more important
- Test Progress report intended for testers and dev would include detailed information about defect types and trends

Communicating the status of testing

Configuration management

- ***A discipline applying technical and administrative direction and surveillance to identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, record and report change processing and implementation status, and verify that it complies with specified requirements***
- It allows testers to manage their testware and test results using the same configuration management mechanism, as if they were as valuable as the source code and documentation for the system itself
- Configuration management supports the build process which is essential in delivery of test release into the test environment
- Configuration management allows us to map what is being tested to the underlying files and components that make it up

Risk management

- ***A factor that could result in future negative consequences***
- ***Risk level is The measure of a risk defined by risk impact and risk likelihood***
 - Product risks
 - Project risks

Product risks

- ***A risk impacting the quality of a product***

- Software might not perform its intended functions according to the specifications
- Software might not perform its intended functions according to user, customer, stakeholder needs or expectations
- A particular computation may be performed incorrectly in some circumstances
- A loop control structure may be coded incorrectly
- Response time may be incorrect for a high performance transaction processing system
- User experience feedback might not meet product expectations

Project risks

- ***A risk that impacts project success***
 - Project issues
 - Organizational issues
 - Political issues
 - Technical issues
 - Supplier issues

Risk management

- ***The process for handling risks***

- Mitigate
 - Take steps in advance
- Contingency
 - Plan B to reduce the impact
- Transfer
 - Transfer the risk to others
- Ignore
 - Do nothing. This is when we know that we cannot do anything now and risk is low

Risk management activities

- Identifying and analyzing what might go wrong
- Determining the priority of risks
- Implementing actions to mitigate
- Having contingency plan if risks occur

Risk based testing

- ***Testing in which the management, selection, prioritization, and use of testing activities and resources are based on corresponding risk types and risk levels***
 - Used risk to prioritize and emphasize appropriate test during test execution
 - Risk based testing start early in the project identifying risks to system quality and using that knowledge of risk to guide test planning, specification, preparation and execution
 - Risk based testing involves both mitigation and contingency
 - Risk based testing also involves measuring how well we are doing at finding and removing defects in critical areas
- Steps
 - Risk analysis
 - Assigning risk level
 - Mitigation options

Risk analysis

- One technique to risk analysis is a close reading of requirement specification ,user story design specification user documentation and other
- Another technique is brainstorming
- Another is a sequence one-to-one or small group of sessions with business and technology experts in the company
 - You could consider risks in the areas of functionality, localization, usability, reliability, performance and supportability
 - You might have a checklist of typical or past risks to be considered
 - You might also want to review the tests that failed and bugs that you found in previous release or similar product

Assigning risk level

- After identifying the risk items, you and if applicable, the stakeholders should review the list to assign likelihood of problems and the impact of problems associated with each one
- You can have business people determine impact and technical people determine likelihood
- Now give a score for both impact and likelihood on a scale of 5 (very high, high, medium, low, very low) and multiply them and put them in table

Product risk	Likelihood	impact	Risk priority number	Mitigation
Risk 1	2	1	2	
Risk 2	4	2	8	

Usage of product risk analysis

- To determine test technique to be used
- To determine particular level and test types to be performed
- To determine extent of testing to be carried out for particular level and type of testing
- To prioritize testing in order to find most critical defect as early as possible
- To determine any activity in addition to testing can be employed to reduce risk such as providing training in design and testing to inexperienced developers
- 2 Tips
 - Remember to consider both likelihood and impact
 - With early risk analysis we make educated guesses, so it's very important to revisit risk analysis at different milestones of testing. In Agile may be every sprint planning

Defect Management

- ***The process of recognizing, recording, classifying, investigating, resolving and disposing of defects***
- Cause of defects include
 - Incorrect software
 - Misconfiguration
 - Failure of test environment
 - Corrupted test data
 - Bad tests
 - Incorrect expected results
 - Tester mistakes

Defect detection percentage

- Metric to find the quality of testing

$$\frac{\text{defects by tester}}{\text{defects by testers} + \text{defects in field}}$$

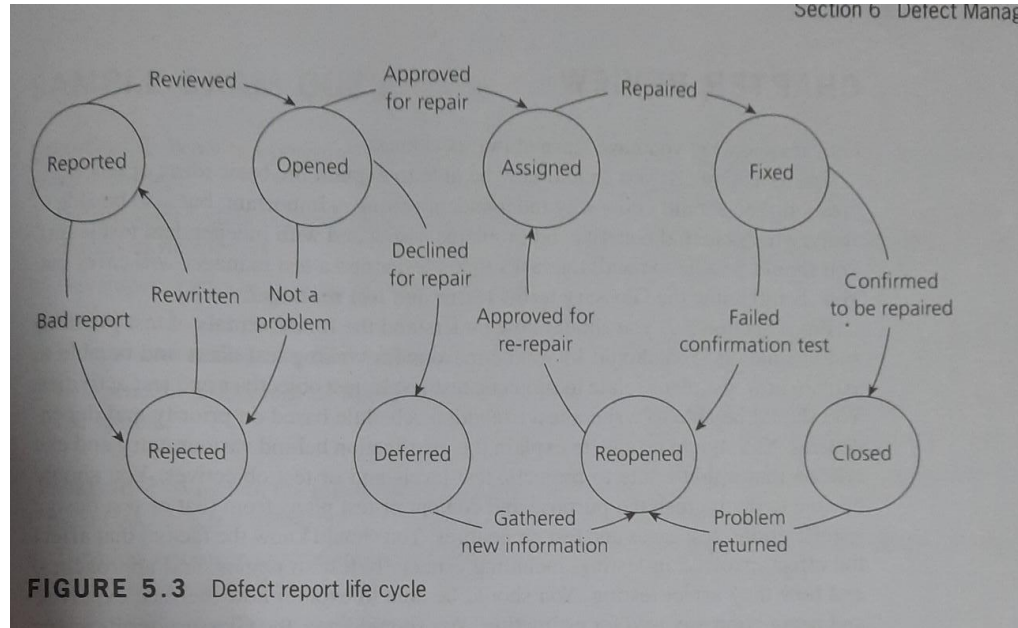
Objectives for defect report

- To provide Developers, managers and others with detailed information about the behaviour observed
- To support test managers in the analysis of trends in aggregate defect data either for understanding more about a particular set of problems or tests or for understanding and reporting the overall level system quality
- To enable defect reports to be analyzed over a project or even across projects to given information and ideas that can lead to development and test process improvement

Best practices for defect report

- An identifier (defect number)
- A title and short summary
- Date of the defect report, issuing organization date and time of failure, name of the tester or author of the defect report
- Identification of test item, test environment, additional information about configuration of the software
- SDLC activity or sprint in which defect was detected
- Description of defect to enable reproduction and resolution such as the steps to reproduce and isolation steps tried
- Expected and Actual results
- Degree of impact of the defect
- Priority
- State of the defect
- Conclusion, recommendation and approvals
- Global issues such as other areas that may be affected by a change resulting from the defect
- Change history
- Reference

Defect life cycle



Test Tools

Test Tools

- ▶ Test tools can be used to support one or more testing activities. Such tools include:
 - ▶ Tools that are directly used in testing, such as test execution tools and test data preparation tools
 - ▶ Tools that help to manage requirements, test cases, test procedures, automated test scripts, test results, test data, and defects, and for reporting and monitoring test execution
 - ▶ Tools that are used for investigation and evaluation
 - ▶ Any tool that assists in testing (a spreadsheet is also a test tool in this meaning)

Advantages

- ▶ Test tools can have one or more of the following purposes depending on the context:
 - ▶ Improve the efficiency of test activities by automating repetitive tasks or tasks that require significant resources when done manually (e.g., test execution, regression testing)
 - ▶ Improve the efficiency of test activities by supporting manual test activities throughout the test process
 - ▶ Improve the quality of test activities by allowing for more consistent testing and a higher level of defect reproducibility
 - ▶ Automate activities that cannot be executed manually (e.g., large scale performance testing)
 - ▶ Increase reliability of testing (e.g., by automating large data comparisons or simulating behavior)

- ▶ **Tool support for management of testing and testware**

- ▶ Test Management Tools
- ▶ Requirements Management Tools (e.g., traceability to test objects)
- ▶ Defect Management Tools
- ▶ Configuration Management Tools
- ▶ Continuous Integration Tools (D)

- ▶ **Tool support for static testing**

- ▶ Tools that Support Reviews
- ▶ Static Analysis Tools(D)

▶ **Tool support for test design and implementation**

- ▶ Test Design Tools
- ▶ Test Data Preparation Tools
- ▶ Model-Based Testing Tools
- ▶ Acceptance Test Driven Development(ATDD) & Behavior Driven Development (BDD) Tools
- ▶ Test Driven Development (TDD) Tools (D)

▶ **Tool support for test execution and logging**

- ▶ Test Execution Tools
- ▶ Coverage Tools (Eg. Requirement Coverage, Code Coverage (D))
- ▶ Test Harnesses (D)
- ▶ Unit Test Framework Tools (D)

- ▶ **Tool support for performance measurement and dynamic analysis**

- ▶ Performance Testing Tools
- ▶ Monitoring Tools
- ▶ Dynamic Analysis Tools(D)

- ▶ **Tool support for specialized testing needs**

- ▶ Data Quality Assessment
- ▶ Data Conversion and Migration
- ▶ Usability testing
- ▶ Accessibility testing
- ▶ Localization testing
- ▶ Security testing
- ▶ Portability testing

Benefits and Risks of Test Automation

- ▶ Potential benefits of using tools to support test execution include:
 - ▶ Reduction in repetitive manual work (e.g., running regression tests, environment set up/tear down tasks, re-entering the same test data, and checking against coding standards), thus saving time
 - ▶ Greater consistency and repeatability (e.g., test data is created in a coherent manner, tests are executed by a tool in the same order with the same frequency, and tests are consistently derived from requirements)
 - ▶ More objective assessment (e.g., static measures, coverage)
 - ▶ Easier access to information about testing (e.g., statistics and graphs about test progress, defect rates and performance)

- ▶ Potential risks of using tools to support testing include
 - ▶ Expectations for the tool may be unrealistic (including functionality and ease of use)
 - ▶ The time, cost and effort for the initial introduction of a tool may be under-estimated (including training and external expertise)
 - ▶ The effort required to maintain the test assets generated by the tool may be under-estimated
 - ▶ The tool may be relied on too much (seen as a replacement for test design or execution, or the use of automated testing where manual testing would be better)
 - ▶ Version control of test assets may be neglected

- ▶ Relationships and interoperability issues between critical tools may be neglected, such as requirements management tools, configuration management tools, defect management tools and tools from multiple vendors
- ▶ The tool vendor may go out of business, retire the tool, or sell the tool to a different vendor
- ▶ The vendor may provide a poor response for support, upgrades, and defect fixes
- ▶ An open source project may be suspended
- ▶ A new platform or technology may not be supported by the tool
- ▶ There may be no clear ownership of the tool (e.g., for mentoring, updates, etc.)

Special Consideration for Test Execution Tools

- ▶ A **data-driven testing** approach separates out the test inputs and expected results, usually into a spreadsheet, and uses a more generic test script that can read the input data and execute the same test script with different data. Testers who are not familiar with the scripting language can then create new test data for these predefined scripts.
- ▶ In a **keyword-driven testing** approach, a generic script processes keywords describing the actions to be taken (also called action words), which then calls keyword scripts to process the associated test data. Testers (even if they are not familiar with the scripting language) can then define tests using the keywords and associated data, which can be tailored to the application being tested.
- ▶ Test management tools often need to interface with other tools or spreadsheets for various reasons, including:
 - ▶ To produce useful information in a format that fits the needs of the organization
 - ▶ To maintain consistent traceability to requirements in a requirements management tool
 - ▶ To link with test object version information in the configuration management tool

Reference

- Foundations of Software Testing: ISTQB Certification by Graham, Black, Veenendaal - [Click to Buy](#)
- TM SQUARE youtube Channel by Neeraj Sir
 - Channel - <https://www.youtube.com/@TMSQUARETECH>
 - His Book - [Click to buy](#)