Session1

**What is Quality?**

Quality is defined as "meeting the requirements of the customer."

**Key Aspects:**

* Good Design
* Good Functionality
* Reliable
* Consistency
* Durable
* Value For Money \*

ISO Definition:

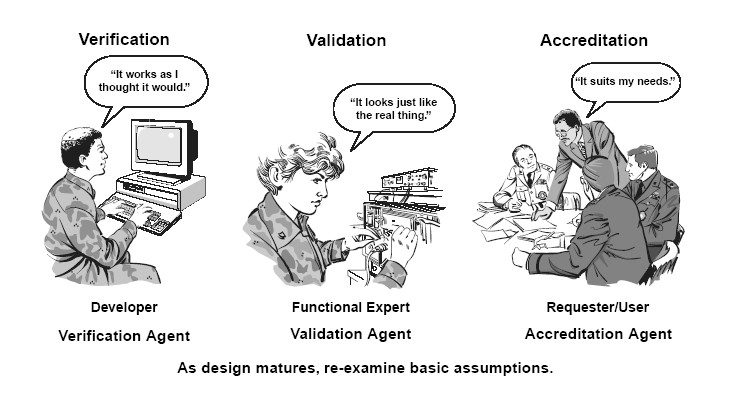
Totality of features & characteristics of a product or service that bears its ability to satisfy customer needs.

**What is Testing?**

Process of evaluation of a Software to detect difference between given input & expected output.

We follow the test process mainly to assess features & quality of a product or project.

Verification & Validation:



***Verification:*** Process to make sure product satisfies / behaves the way we want (delivers all functionality required)

***Validation:*** Process to make sure product is built as per customer requirements (test as per requirements)

\*\*\*Difference between Error, Defect, Bug, Fault & Failure:

****

**Error:** Mistake in code

**Defect:** Mismatch in application’s behavior due to error (mostly found by tester)

**Bug:** Defect accepted by Development team

**Fault:** Any Incorrect step which causes unexpected behavior in application

**Failure:** Product that does not meet the minimum specified requirements

**What is Software Quality?**

According to ISO, software quality is defined as ‘Degree of conformance to explicit & implicit requirements’

Explicit requirements: Components, features, functionality specified by customer

Implicit requirements: Performance, Usability, Scalability, Portability etc.

According to ISTQB, it is defined as the ‘Degree to which system meets requirements’

**Why Testing?**

Because, we all make mistakes….. Someone else (other than dev team) is required to spot flaws in the product.

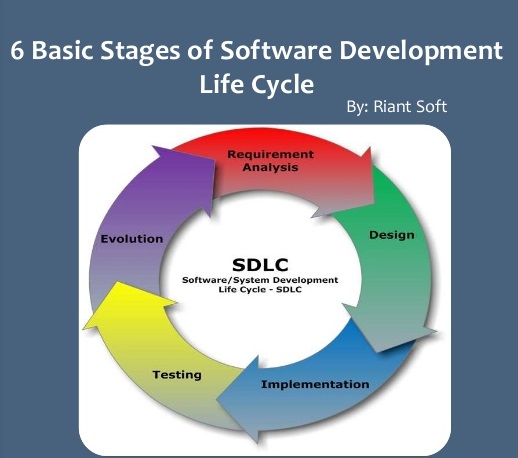
\*\*\*Not following the test process might lead to:

Loss of money

Loss of time

Damage to business

**Software Development Process:**



**What is Software Quality Assurance?**

Process that ensures developed software meets with defined (or) standard quality specifications…

It is an ongoing process in SDLC. Testing is required in each phase of the product development life cycle, rather than at end.

**What is Software Quality Control?**

Set of activities to ensure quality is achieved in the developed product/ service.

Ex: Reviews (Requirement/ Design/ Code/ Plan/ Test Case review)

Testing (Unit/ Integration/ System/ Acceptance etc…)

**What are Objectives of Quality Assurance/ Control?**

1. Customer Focused
2. Leadership
3. Involvement
4. Process Approach
5. Continual Improvement

**What is ‘Cost of Quality’?**

Cost of NOT creating a quality product & failing to meet the customer requirements.

The main cost would be reflected in below areas:

* Review
* Planning
* Training

**What are Software quality factors?**

* Understandability (Purpose)
* Completeness (Fully Developed)
* Portability
* Well – Documented
* Testability
* Reliability

Session2

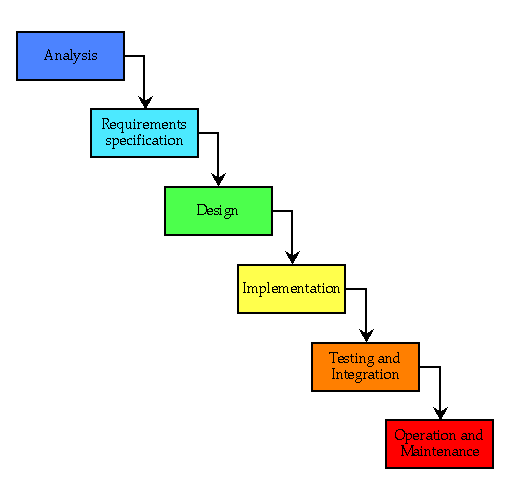
Software Development Life Cycle:

****

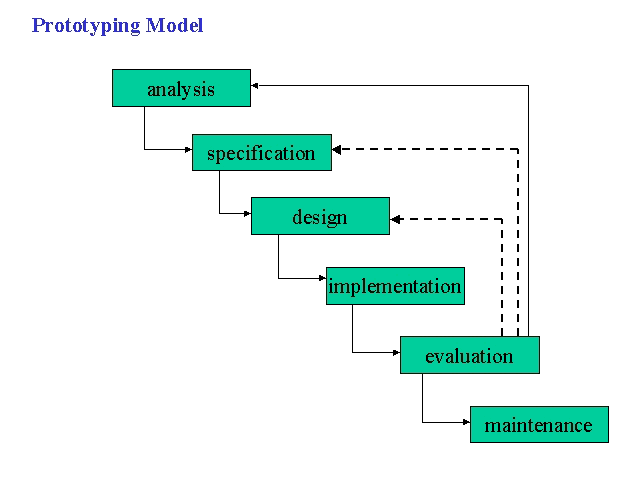
* **Requirements Gathering** (Business Requirements Specification)
* **Requirements Analysis** (Software/ Functional Requirement Specifications)
* **Design** 
  + **High - Level Design** (System)
  + **Low- level design** (Components/ Modules)
* **Development** (Code)
* **Testing** (Unit & System)
* **Deployment**
* **Support & Maintenance**

**SDLC Engineering Models:**

Waterfall model (Linear Sequential Model):

****

Prototyping model:

****

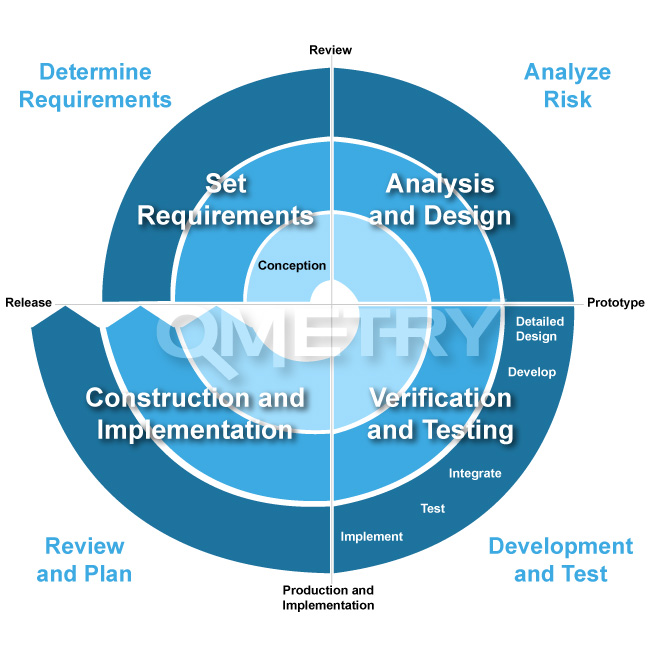
Advantages:

* User gets a proper clarity & feel of functionality
* Mainly used for Non-IT people
* Helps to demonstrate the concept
* Reduces risk of failure
* Time required to complete the project after getting final SRS reduces

Disadvantages:

* Prototyping is done at cost of developer
* Prototype is of no use, after original SRS is given
* Slow Process
* Too much involvement of customer

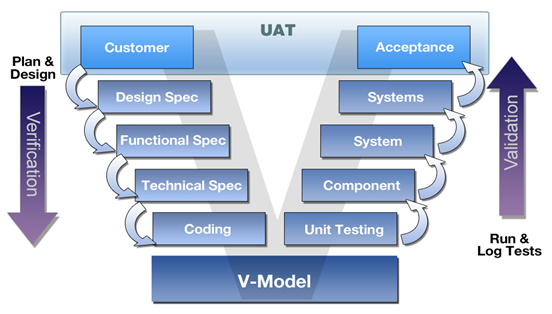
Spiral Model:



This model uses many of the same phases as the Waterfall model separated by planning and risk assessment.

* Projects
* Releases
* Cycles
* Phases

V- Model:



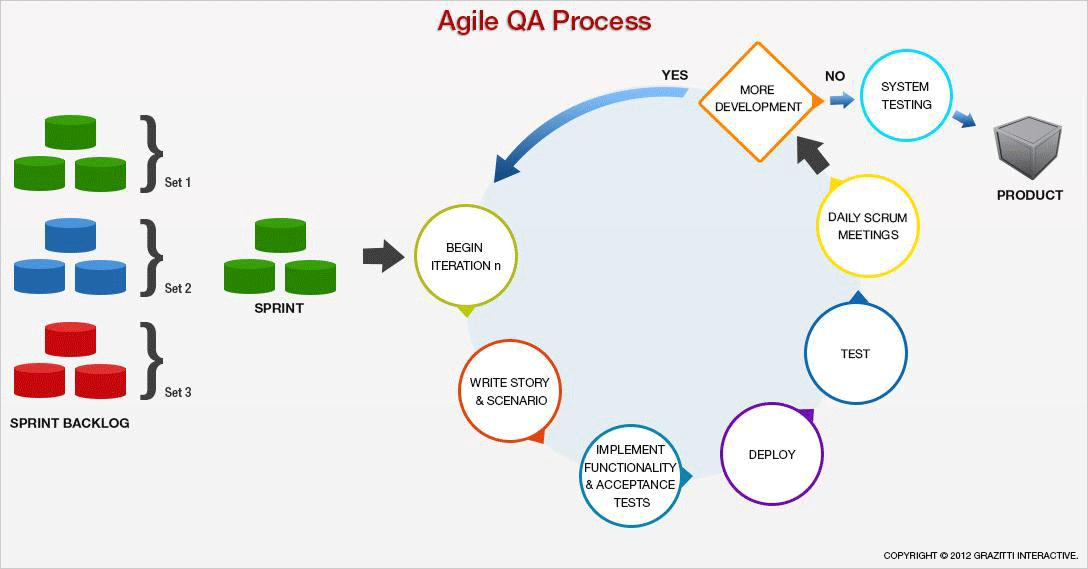
It is a process where development & testing can do parallel. For every development phase, there is a testing phase.

Development phases are called as verifications & testing phases are called as validations

Here the verification means the software implements correctly or not & the validation means the software that has been built is traceable to customer requirements or not.

Agile Process:

It is a type of incremental model. Extreme Programming (XP) is well known agile life cycle model.



Advantages:

* Customer Satisfaction by continuous delivery
* Continuous improvement

Disadvantages:

* Difficult to assess the effort required at beginning of SDLC
* Project can be taken off track if customer is not clear what final outcome they want

**Software Test Life Cycle Process:**

* Test Strategy
* Test Planning
* Test Design (Scenarios/ Cases)
* Test Execution
* Defect Reporting & tracking
* Status Reporting
* Test Closure

Session3

**Quality Standards:**

**ISO:** ISO International Standards ensure that products and services are safe, reliable and of good quality.

**CMMI:**

Level1: Initial – Process poorly controlled

Level2: Managed – Process characterized for projects

Level3: Defined – Process characterized for the organization

Level4: Quantitatively Managed – Process measured and controlled

Level5: Optimizing – Focus on continuous process improvement

**Requirements Management:**

****

The most common cause of requirement management failure is ‘**inadequate’** & ‘**inefficient’** management of changes.

* Uncontrolled Changes
* Static Documentation

Failure due to poor requirement management:

* Confusion
* Missed Schedules
* Cost Overrun
* Unsatisfied Users & developers

Top Reasons for a well-functioning requirements management:

* Having an updated requirements repository
* Manage requirements by release, risk etc.
* Follow up & communicate changes
* Ensure team members access to requirements
* Tracking status of requirements
* Define test cases according to the requirements

**Software Testing Concepts:**

*Testing Methodologies:*

Static Testing (Verification):

Code is not executed. Manual verification of code, requirement documents, design documents to find errors.

This is done during the initial phase of testing.

*Objective:*

To improve the quality of software product by finding errors in early stages of development cycle.

*Work documents:*

Requirement specifications

Design Document

Source Code

Test Plans

Test Cases

Test Scripts

*Techniques:*

Informal Reviews

Technical Review

Inspection

Code Review

Dynamic Testing (Validation):

Code is executed. Checks the functional behavior of software or system.

*Objective:*

To confirm that the software product works in conformance with the business requirements.

*Techniques:*

Unit Testing

Integration Testing

System Testing

**

|  |  |
| --- | --- |
| Static Testing | Dynamic Testing |
| Testing done without executing the program | Testing done by executing the program |
| Verification | Validation |
| Prevention of defects | Finding & Fixing the defects |
| Assessment of code & documentation | Bugs/ Bottlenecks in application |
| Involves checklist & process to be followed | Involves test cases for execution |
| Can be performed before compilation of code | Performed after compilation of code |
| Cost of finding defects & fixing is less | Cost of finding defects & fixing is high |
| High return on investment | Low return on investment as it is after development |
| More Review comments for good quality | More Defects for good quality |

Session4

Black Box Testing:

Testing without any knowledge of interior working of application.

Advantages:

* Code access is not required
* Separates users & developer’s perspective of application
* No knowledge of implementation/ programming language is required

Disadvantages:

* Limited coverage of testing
* Inefficient due to the limited knowledge about application
* Blind Coverage, cannot target error prone areas

White Box Testing (Glass/ Open Box):

Detailed testing of internal logic & code structure.

Advantages:

* Easy to find the code which causes hidden defects
* Maximum test coverage

Disadvantages:

* Skilled tester is needed, which leads to cost increase
* Tools are required

Gray Box Testing:

Test application with limited knowledge of internal workings of application.

Advantages:

* Combined benefits of Black & White box
* Testing is done from users point of view

Disadvantages:

* Takes more time to test

Levels of testing:

* Unit Testing:
  + Testing performed by developers before the setup is handed over to the testing team to formally execute test cases
  + **Goal:** To isolate each part of program and show that individual parts are correct in terms of requirements & functionality
  + **Limitation:** It is impossible to evaluate every line of code
* Module Level Testing:
  + Testing performed for independent modules before they are integrated as system
  + Saves time & effort by validating module level requirements even before the system testing happens.
* Integration Testing:
  + Testing of combined parts of an application to determine if they function correctly together.
  + **Bottom – Up Testing:** Begins with unit testing, followed by higher level of combinations of units/modules.
  + **Top- Down Testing:** Highest level of modules is tested first, and progressively lower – level modules are tested after that.
* System Testing:
  + Tests the system as whole
  + Application as whole is tested rigorously to see that it meets quality standards
  + Both Functional & Non-functional requirements are validated
  + Application tested in an environment which is very close to the production environment where the application will be developed.
* Regression Testing:
  + Whenever there is a change is made in application it is possible that other areas within application have been affected by this change.
  + To verify that a fixed bug has not resulted in another functionality violation is regression testing
* User Acceptance Testing:
  + To test how the application will perform in production environment.
  + **Alpha Testing:** First stage of acceptance testing which is performed in Developers site in customer presence.
  + **Beta Testing:** Testing which is performed after Alpha testing has been successfully performed.
    - Limited people will attend this testing
    - Users will install, run the application and send their feedback to project team
    - Using the feedback, project team can fix the problems before releasing to actual users

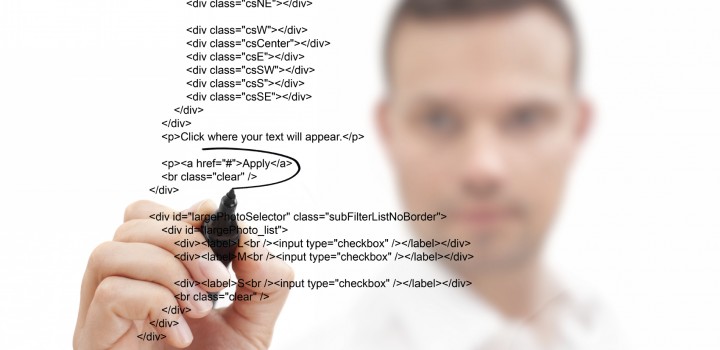
**Static Testing:**

* Walkthroughs:
  + It is not a formal process/ review
  + Main goal is to examine and discuss the validity of the proposed solutions
* Reviews:
  + Main purpose is to monitor progress, status and make decisions about future actions
  + Checks consistency with the deviations from plan
  + Assessment of project risks
  + Design review
  + Outcome includes action items and issues to be resolved
* Inspections & Audits:
  + Check whether the process is followed as per standards or not
  + Takes much less time (2 to 3 hrs.)

Session5

**White Box Testing and Designing techniques:**

* Code Review:

****

* Verify Duplicate Code
* Incorrect Exception handling (Catch exception with multiple catches)
* Hardcoding in code should be avoided
* Add proper comments to code
* Integration Testing:
* Top – Down Integration Testing
* Bottom – Up Integration Testing
* Big Bang Integration Testing (Program is integrated without any formal integration testing)
* Stub: Dummy module that always simulates the low level modules when sub programs are under construction (Top – to – Bottom testing)
* Driver: Dummy modules which always simulate the high level modules
* Mutation testing:
* Introducing small errors (mutations) into your application to see if test suite picks it up or not.
* If test suite does not pick up the mutations, it is deficient and should have more test cases added.
* Mutation testing tests your test suite rather than your application

**White Box Testing design:**

* Statement Coverage:
* Make sure that all / important code logic is notified
* Flow Graphs & Path testing:
* All possible flows in the code are covered in test design
* Cyclomatic Complexity:
* All possible flows in the code are covered in test design
* Data Flow Testing:
* Critical Data flowing between integrated modules is handled
* Syntax Testing:
* Check the code syntax for compilation errors

Session6

Black Box Testing:

* Behavioral Testing:

Black box testing is also called as behavioral testing where we are verifying the behavior of application under different circumstances

* Functional & Non Functional Testing:

Functional Testing: Validating the product meets functional requirements

Non-Functional Testing: Validating the product meets non- functional requirements too.

* BVT/ BAT:

Build Verification Testing: Set of tests run on every new build to verify that the build is testable before it is released to test team for further testing.

These test cases are core functionality test cases that ensure application is stable and can be tested thoroughly.

If BVT fails, the build is again assigned back to DEV team for immediate fix.

BVT is also called ‘**Smoke Testing**’ or ‘**Build Acceptance Testing**’.

Advantage: Saves the effort of a test team to setup & test when a major functionality is broken.

Which test cases should be included in BVT?

* Critical test cases
* Stable
* Test coverage
* Ad Hoc Testing:

Informal approach to assess a product. Testers to a random & monkey testing.

Advantage: Flaws can be attended to more quickly than if they had to be approached in a more systematic fashion.

* System Testing:

Testing of a complete fully integrated software product based on SRS to test system as a whole.

* End-to-End Testing:

To test whether the flow of an application is performing as designed from start to finish.

Main purpose is to identify system dependencies and to make sure that the right information is passed between various components.

* Sanity Testing:

Narrow regression testing that focus on one or few areas of functionality.

Narrow & deep testing.

* Regression Testing:

Selective retesting of a software system that has been modified to ensure that any bugs has been fixed / any new functionality added is fine along with testing of the working functionality has any impact because of that.

Automation tools are used mainly to do the regression testing.

* Alpha Testing:

Testing performed by the developers/ testers at the software development site.

* Beta Testing:

Testing performed by the customers at their own site.

Differences between Alpha & Beta Testing:

|  |  |
| --- | --- |
| Performed by the developers at their site | Performed by the customers at their site |
| Sometimes also performed by testing team | Not performed by testing team |
| Not open to public | Open to the public |
| Performed in virtual environment | Performed in real environment |
| Performed within the organization | Performed outside the organization |

* Acceptance Testing:

Testing done to establish the confidence in system

* Performance Testing:

Testing done to check how the system is performing in a given particular situation.

Resource usage, scalability and reliability are validated.

Goal: is to establish benchmark behavior of system.

* Load Testing:

To test the system constantly and steadily by increasing the load on system till the time it reaches the threshold limit.

Goal: to identify defects in application related to memory leaks.

* Stress Testing:

Negative testing done to break down the system.

Goal: to define the behavior of application after failure.

* Usability Testing:

To test the ease with which user interfaces can be used.

To check whether the product is user friendly or not.

* Install/ Uninstall Testing:

To test how easily can the installation/ uninstallation is done?

* Recovery Testing:

Testing is done in order to check how fast & better application can recover after it has gone through any crash or hardware failure etc.

It is a forced failure of software.

* Security Testing:

Testing done to check whether application is secured or not.

* Compatibility Testing:

Tested to ensure compatibility of system built with various other browsers/ OS/ platforms/users etc..

* Comparision Testing:

Comprises of comparing the software product’s strengths and weaknesses with other software that are currently available in market.

Session7

Product/Project Environments:

* Technical / Development Environment:

Environment where you will create, configure & customize the image of an application which is to be promoted to another environment.

* Test Environment:

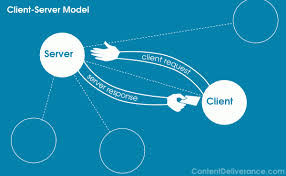
Environment where the testing team tests application using test data (not real data).

This environment closely simulate the production environment and intended users tests the application.

* Production Environment:

Environment where the application is deployed for intended users. It is a real time environment for organization or commercial daily operations.

* Client / Server Environment:



Environment where client (applications) contacts servers (systems that gives back the required information) & servers give the response back.

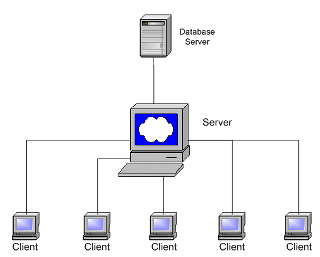
Important information shared like: Time, Files, Connection etc…

Advantages:

Centralized resources

Improved security

* Client / Server Architecture:



Application Server

DB Server

* Web Application:

3 Tier applications (Presentation Layer/ Business Layer / Database)

* Database Servers:

Backend system of a database application using client/ server architecture.

Performs tasks such as data analysis, storage, manipulation etc..

Ex: SQL Server, Oracle

* Web Servers:

Computers that serves requests from web pages.

* Application Servers:

Program that handles all application operations between users and backend business.

Ex: Weblogic, tomcat etc..

* VPN Servers:

Point – to – Point connections across public or private network.

* Java Environment:

Object Oriented Programming developed using Java language.

Mostly web applications & Applets.

* Dot Net Environment:

Object Oriented Programming developed using .Net language

May be web or windows applications

* ERP or SAP environments:

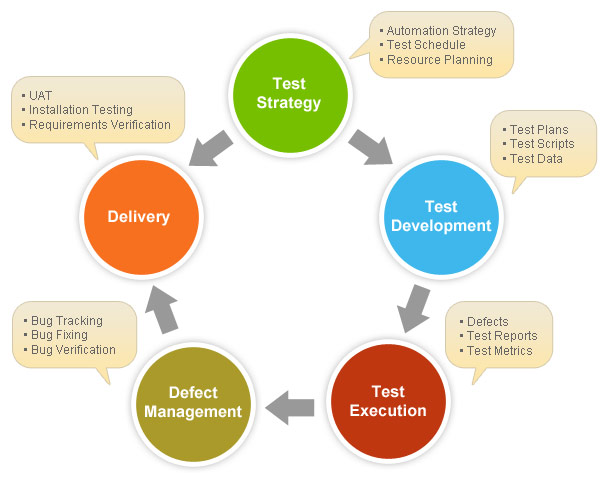
Industry term for the broad set of activities that helps business manages important parts of business.

* Unix Environment:

Only File manipulation is possible. Supported by Linux Operating system.

Session8

**Test Strategy & Planning:**

****

* Overview to Test Strategy:
  + To make a clear plan of approach for achieving the test objectives.
  + One of the most important documents for QA team.
  + Initiates thought process which helps to discover many missing requirements.
  + Thinking and test planning activities help team to define testing scope and test coverage.
  + It helps test managers to get clear state of the project at any point.
  + The chances of missing any test activity are very low when there is a proper test strategy in place.
  + Test strategy defines guidelines for test approach to be followed in order to achieve the test objectives and execution of test types defined in testing plan
  + It deals with test objective, approach, test environment, automation strategy and tools, and risk analysis with contingency plan.
* Test Strategy document:

1. Scope & Overview
   1. Project Overview
   2. Testing Activities
   3. Timelines
2. Test Approach
   1. Testing process
   2. Level of Testing
   3. Roles & Responsibilities
   4. For all Test Case Types
      1. When to Start
      2. Test Owner
      3. Automation Strategy & Tool
3. Test Environment
   1. Development
   2. Test & UAT
   3. Data handling
4. Testing Tools
5. Release Control
6. Risk Analysis
7. Review Approvals

* Approach for Testing:

1. Analytical
2. Process
3. Dynamic
4. Regression

* Test Issues:

1. Confused Test Team
2. Test maintenance failure
3. Manual Testing
4. Selecting the right tests

* Risk Analysis:

1. Which functionality is most important?
2. Which functionality is most visible to the user?
3. Which functionality has the largest safety impact?
4. Which functionality has the largest financial impact on customer?
5. Which parts of code are more subject to error?
6. Which features had caused problems in previous projects?
7. What do the developers think are the highest- risk aspects of application?
8. What kind of problems would cause worst publicity?
9. What kinds of problems would cause the most customer service complaints?

* Test Plan:

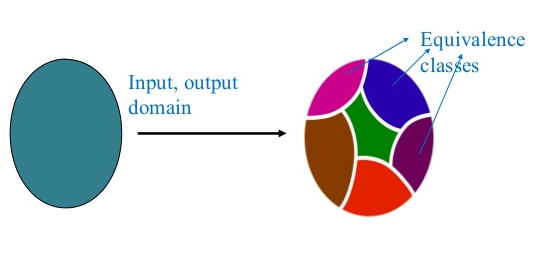
1. Introduction
   1. Test Plan Objectives
2. Scope
3. Test Strategy
   1. System Test
   2. Performance Test
   3. Security Test
   4. Automated Test
   5. Stress Test
   6. Recovery Test
   7. Documentation test
   8. Beta Test
   9. User Acceptance Test
4. Environment Requirements
5. Test Schedule
6. Control Procedures
   1. Reviews
   2. Bug Review Meetings
   3. Change Request
   4. Defect Reporting
7. Functions to be tested
8. Functions not to be tested
9. Resources & Responsibilities
10. Deliverables
11. Suspension / Exit Criteria
12. Test Estimation
13. Risks
    1. Schedule
    2. Technical
    3. Management
    4. Personnel
    5. Requirements
14. Tools
15. Documentation
16. Approvals

Session9

Test Cases Design Methods & Techniques:

**Specification based Test Design Techniques (Black Box Testing):**

* Equivalence Class Partitioning:



* + Divide all possible inputs into classes (partitions) such that
    - Program behaves same for inputs in the same class
    - Test with a representative value from a class is sufficient
    - If representative detects fault then other class members will detect the same fault
  + 2 types: Valid & Invalid classes

Example1:

If a student has total score >=35, he passes the exam. Otherwise, he fails.

**Valid Equivalence Class Invalid Equivalence Class**

Total Score 1. >=35 2. <35

3. Null

4. String (Absent)

Result 5. Pass

6. Fail

7. Error

**Possible Test Cases in above conditions:**

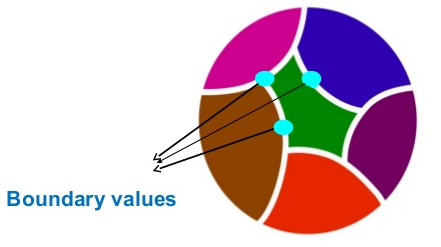
TC1: 1, 5

TC2: 2, 6

TC3: 3, 7

TC4: 4, 7

* Boundary Value Analysis:



* + Testing the boundary conditions of all equivalence classes defined
    - i.e. values directly on, above, and beneath the edges of classes defined

Example:

If a student has total score >=35, he passes the exam. Otherwise, he fails.

**Valid Equivalence Class Invalid Equivalence Class**

Total Score 1. >=35 2. <35

3. Null

4. String

Result 5. Pass

6. Fail

7. Error

**Possible Test Cases in above conditions: Data to Test**

TC1: 1, 5 TC1a: 35, Pass

TC1b: 36, Pass

TC2: 2, 6 TC2: 34, Fail

TC3: 3, 7 TC3: Null, Error

TC4: 4, 7 TC4: String, Error

**Example for both Equivalence & Boundary class:**

An Input box accepts values from 1 to 1000.

Equivalence Partitioning:

1. Class with value range from 1 to 1000
2. Class with values below lower limit (1)
3. Class with values above upper limit (1000)

Boundary Value Analysis:

1. Exact Values 1 and 1000
2. Values just below the input domain (0 and 999)
3. Values just above the input domain (2 and 1001)

* Decision Tables:
  + Represents all possible conditions & actions associated with the conditions

Example:

(a) If the salary of a person is less than equal to Rs. 70,000 and expenses do not exceed Rs. 30,000 then 10% tax is charged by IT department.

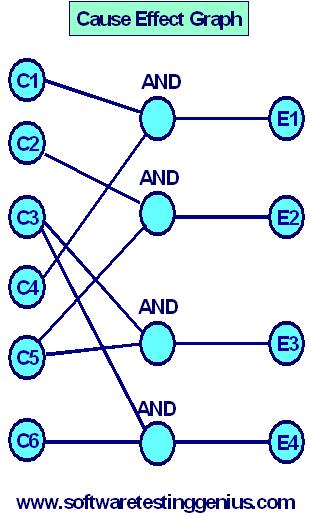
(b) If the salary is greater than Rs.60,000 and less than equal to Rs 2lakhs and expenses don't exceed Rs. 40,000 than 20% tax is charged by IT department.

(c) For salary greater than Rs 2 lakhs, 5% additional surcharge is also charged. (d) If expenses are greater than Rs. 40,000 surcharge is 9%.

Step1: Identify the Causes & its Effects:

|  |  |
| --- | --- |
| **Causes** | **Effects** |
| C1 : Salary < = 70,000  C2 : Salary > 60,000 and Salary < = 2 lacs  C3 : Salary > 2 lacs  C4 : Expenses < = 30,000  C5 : Expenses < = 40,000  C6 : Expenses > 40,000 | E1 : 10% tax is charged.  E2 : 20% tax is charged.  E3 : (20% tax) + (5% surcharge) is charged.  E4 : (20% tax) + (9% surcharge) is charged. |

Step2: Draw the Cause & Effect graph:



Step3: Transform the Cause- Effect graph into a decision table.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | **1** | **2** | **3** | **4** |
| Condition / Cause | C1 | 1 | 0 | 0 | 0 |
| Condition / Cause | C2 | 0 | 1 | 0 | 0 |
| Condition / Cause | C3 | 0 | 0 | 1 | 1 |
| Condition / Cause | C4 | 1 | 0 | 0 | 0 |
| Condition / Cause | C5 | 0 | 1 | 1 | 0 |
| Condition / Cause | C6 | 0 | 0 | 0 | 1 |
| Action / Effect | E1 | X | - | - | - |
| Action / Effect | E2 | - | X | - | - |
| Action / Effect | E3 | - | - | X | - |
| Action / Effect | E4 | - | - | - | X |

Causes & Effects are nothing but the ‘Conditions’ & ‘Actions’ here.

i.e. If C1 & C4 are true (1), then the effect (action) is E1.

Step4:

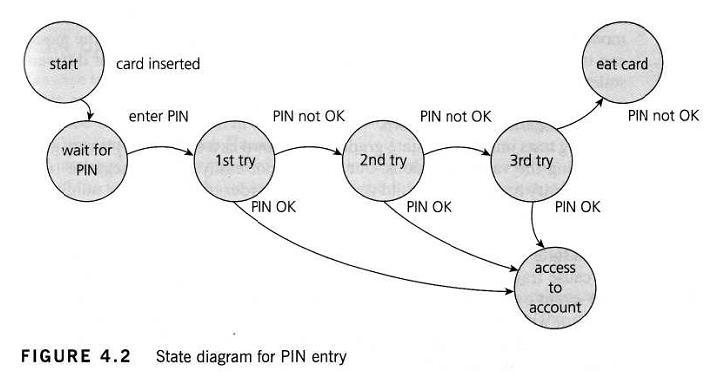
Since there are 4 rules in decision table, there should be at least 4 test cases.

Test cases can be:

1. Salary = 20,000, Expenses = 2000.  
2. Salary = 1,00,000, Expenses = 10,000  
3. Salary = 3,00,000, Expenses = 20,000   
4. Salary = 3,00,000, Expenses = 50,000

* State Transition Testing:

It is applicable when system can be in a number of different states, and transitions are defined by the rules.

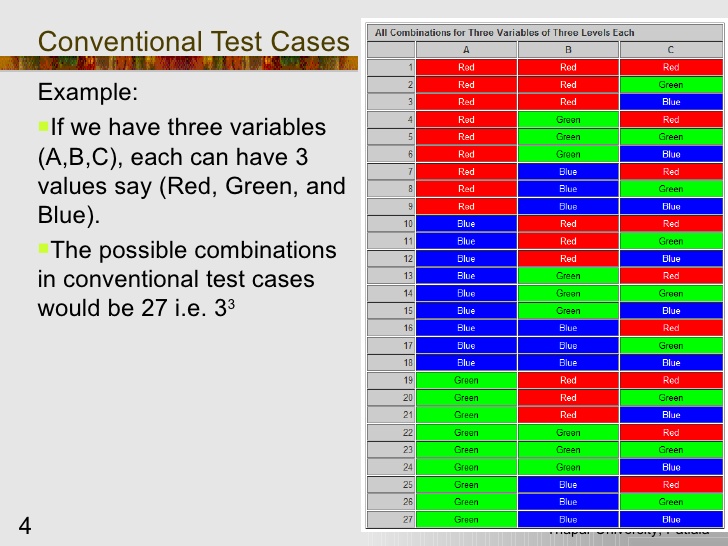


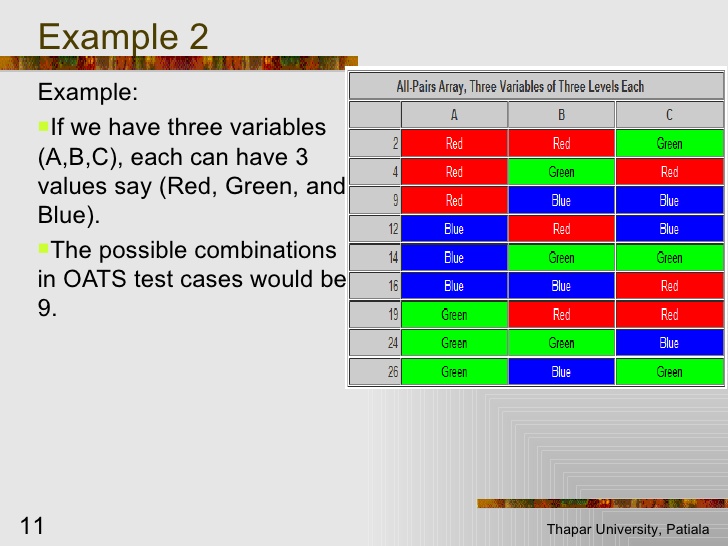
Possible test cases are:

* Normal situation, where PIN is entered correctly for the first time
* Enter incorrect PIN each time and system takes the card in.
* PIN is incorrect for the first time and Ok in the second time
* PIN was correct in the 3rd try
* Navigate to a different transaction

Each state can be noted as a test condition.

* Orthogonal Array Testing:





* All Pairs Testing:

Covers all the possible combinations (Image 1 in above scenario)

Session10

* Error Guessing:

In this technique, past experience of a tester is used to find the components of application where error might happen.

Mostly done by the most experienced testers who can use their past experience to find defects.

It has no rules or certain procedure, only based on tester previous skills.

Example:

1. Value division by Zero
2. Pressing submit button on form without filling any entries
3. Entering all kinds of incorrect data to break the system

**Structure based Test Design Techniques (White Box Testing):**

* Formal Reviews
* Statement Coverage
* Path (Control Flow)
* Data flow
* Syntax check

**Scenario based Test Designing:** Specific to a particular scenario

**Field Validation Based Test Designing:** Validations for a particular field in application

1. Text Box input level validations (Entering positive & negative values etc.)
2. Password field level validations (character combinations, encryption etc.)

**‘Use Case’ Based Test Designing:** For a use case specific to a module.

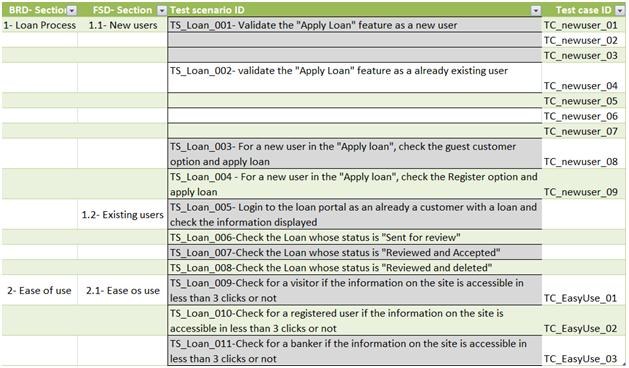
Example: Module – Debit Amount from bank account

Use Cases: Debit from Savings Account

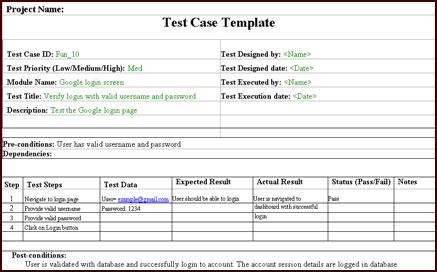
Debit from Current Account

Balance Enquiry

**Traceability Matrix:** Verify the coverage of requirements in test case writing

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2013/10/simple-Traceability-Matrix-2.jpg)

**Test Case Template:**



Session11

**Test Scenario Vs Test Case:**

|  |  |
| --- | --- |
| Test Scenario is nothing but a test procedure | Test Case consists of input values, precondition, expected & actual results developed to cover certain test condition |
| Test Scenario has one – to – many relation with test case | We write test cases for each scenario |
| Test Scenarios are derived from use case | Test Case is derived/ written from test scenario |
| Test Scenario represents a series of actions that are associated together | Test Case represents a single action by user |
| Test Scenario is ‘What to be Tested’ | Test Case is ‘How to be tested’ |

**Test Script:**Set of instructions (written using a scripting/programming language) that is performed on a system under test to verify that the system performs as expected.

Test scripts are used in automated testing.

Test Scripts that are written in human language are called ‘Test Cases’.

**Test Suite:**

Collection of test cases to be executed at certain period of time.

**What is Test Execution?**

Test execution involves running the test cases developed for the system and to report the test results.

Pre-Requisites:

* Test Environment availability
* Test Automation (if planned)
* Test Data availability
* Other applications that interact with the software

**Types of Test Execution:**

Manual (Test Cases): Execution of test cases manually against the application available

Automation (Automated Script): Execution of test scripts in Automation tool against the application available

**Test Data:**

Data that we use during the test execution is called as ‘Test Data’.

Data generation process: Customer’s data, inserting data in Database etc.

**Test Case Execution:**

During the test case execution, we will compare the expected results in test case with the actual result/ behavior identified in application under test.

Status is defined for each test case (Pass/ Fail)

Below Test Metrics are submitted after each execution cycle.

No. of Test Cases Executed / Total No. of test cases

No. of Test Cases Passed

No. of Test Cases Failed

Defects identified (Severity & Priority)

**Test Execution Life Cycle Process:**

Implementation

Code Unit Test Case Execution

Defects Reporting

Application Manual/ Automation Test Case Execution

**Levels of Test Execution:**

Level 0 – Sanity / Test Acceptance / BVT

Level1 – Comprehensive

Level2- Regression

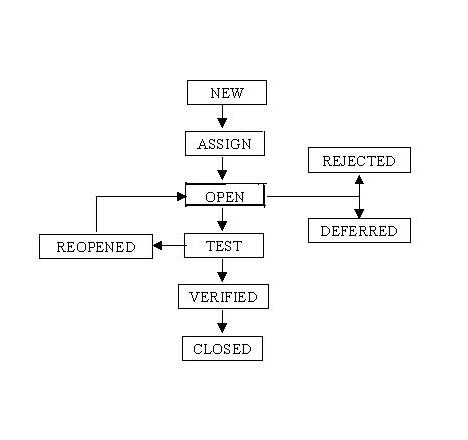
Level3 – Pre-Acceptance

Level4- User Acceptance Test (Alpha & Beta)

Session12

**Defect/ Bug Reporting and Management:**

Defect / Bug Life Cycle:



**New:** When a defect is logged/ posted for the first time.

**Assigned:** After tester posts the bug, test lead/ reviewer approves that the bug is genuine and he assigns it to development team.

**Open:** Developer analyzes the defect.

**Fixed:** Developer makes necessary code changes and verifies the changes & then makes the status as Fixed. Bug is passed to testing team for verification.

**Verified:** Tester again tests the fix in application. If bug is not present in software, status is changed to ’Verified’.

**Reopen:** If Bug still exists even after the bug is fixed by the developer. Entire lifecycle is again repeated from Assign till the rest of the phases.

**Closed:** Once Bug is fixed, tester feels no longer it exists in software. Status is changed to ‘Closed’.

**Rejected:** If the developer feels that the bug is not genuine, he rejects the bug.

**Deferred:** Bug is expected to be fixed in next releases.

**Severity & Priority of a Bug**:

Severity:

The extent to which the defect can affect software. It defines the impact that a given defect has on a system.

Priority:

The order in which we should resolve a defect. The priority is set by the tester to the developer mentioning the time frame to fix the defect.

If High Priority, developer has to fix it at earliest. It depends on the customer requirements too.

Ex:  If the company name is misspelled in the home page of the website, then the priority is high and severity is low to fix it.

Examples for Defect Severity & Priority:

Severity Priority Example

**Low** **High** Spell Mistakes

(UI Related) **Low** Improper Alignment

**Medium High** does not allow valid type

(Boundary) **Low** allows invalid type

(Error Message) **High** does not throw error message

**Low** Improper error message

**High High** Final output is wrong

**(Calculation) Low** Dependent results are wrong

**Writing Bug Description:**

Summary: Brief description of bug in 1 or 2 lines.

Component: Module on which bug is identified

Description: Detailed information of bug.

Steps to recreate: The steps following which bug can be reproduced.

SRS reference: Requirement specification reference, which functionality is related to bug.

Expected Result: Expected outcome of the functionality.

Actual Result: Actual behavior identified in application, during the test execution.

**Categories of Defects:**

Severity Wise:

Major: defect which will cause an observable product failure.

Minor: Defect that will not cause a failure in execution of product

Fatal: Defect which will cause system to crash or close suddenly.

Work Product Wise:

FSD: Defect from Functional Specification Document

DDS: Defect from Detailed Design Document

Source Code: Defect from Source Code

Types Of Errors:

Comments

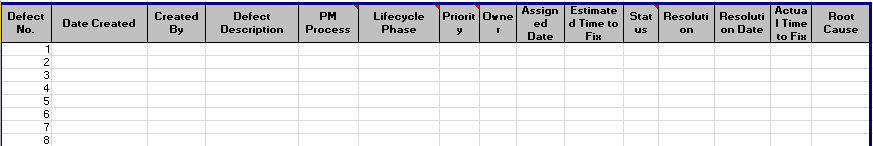
Computational Error: Improper Business code

Data Error: Incorrect data taken during testing

Message Error: Incorrect error messages

System Error: Hardware related error

Defect Reporting Template:



Session13

**Test Measurement:**

To measure the test process progression at any point of time.

**Test Status Reports:**

Weekly/ Monthly status report includes the below details.

1. Modules under testing
2. Number of total test cases executed
3. Number of test cases which are in progress
4. Number of total test cases planned for execution
5. Total Defects identified during testing
   1. Severity
   2. Priority
   3. Status
6. Issues/ Clarifications need to be discussed
7. Execution plan for the next period (Week/ Month)

Test Execution % (Coverage) = Number of units tested / Total size of the system

**Test Sign- Off:**

1. Test Strategy Compliance
2. System Test Plan adherence
3. Requirements Traceability Matrix
4. Manual / Automated Test Scripts completeness
5. Test Execution/ Bug Summary Reports