Manual Testing - Orientation Session

By,

Dayanand Patil

Agenda for discussion

- Introduction to testing
- Principles of testing
- Phases of testing
- Types of system testing
- Static Vs Dynamic testing
- Black Box Vs White Box Testing
- White Box and Black box test case design techniques
- Positive,Negative,Basic and Alternate tests
- Importance of writing positive, negative, basic, alternate test while designing test cases

Agenda for discussion

- Test case execution life cycle
- Defect Life Cycle
- Defect Severity and Defect priority
- Scenarios related to the severity and priority
- Discussion on sample test scenario and test case design
- Discussion on defect report design
- Fun-Kahoot Quiz on covered discussion

Introduction to testing

- Testing is the process of executing a program with the intent of finding errors as early as possible in SDLC.

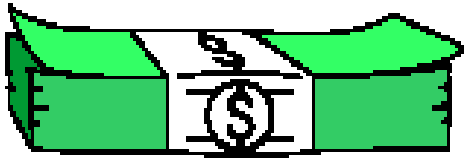
- The process consisting of all life cycle activities both static and dynamic concern with planning, preparation and evaluation of software products and related work products to determine that they satisfy specified requirements, to demonstrate that they are fit for purpose and to detect defects.

Need for Testing

- **Contribute to the delivery of higher quality software product**
- **Undetected errors are costly to detect at a later stage**
- **Satisfied users and to lower maintenance cost**

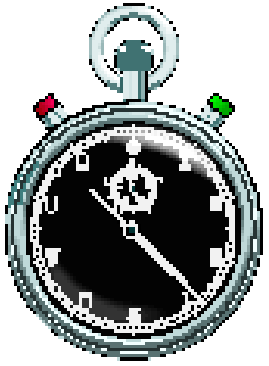
Objectives of Testing

- **To find greatest possible number of errors with manageable amount of efforts applied over a realistic time span with a finite number of test cases**



Cost effective

Time limited



How Testing is conducted

- By examining the users' requirements
- By reviewing the artifacts like design documents
- By examining the design objectives
- By examining the functionality
- By examining the internal structures and design
- By executing code

Principles of Testing

- Exhaustive testing is impossible
- Defect clustering
- Pesticide paradox
- Testing shows presence of defects
- Absence of Error
- Early Testing
- Testing is context dependent

Testing Phases

- **Unit testing**
  - Unit testing is code-based and performed primarily by developers to demonstrate that their smallest pieces of executable code function suitably.
- **Integration testing**
  - Integration testing demonstrates that two or more units or other integrations work together properly, and tends to focus on the interfaces specified in low-level design.
- **System testing**
  - System testing demonstrates that the system works end-to-end in a production-like environment to provide the business functions specified in the high-level design.
- **Acceptance testing**
  - Acceptance testing is conducted by business owners and users to confirm that the system does, in fact, meet their business requirements.

Types of System Testing

Types of System Testing

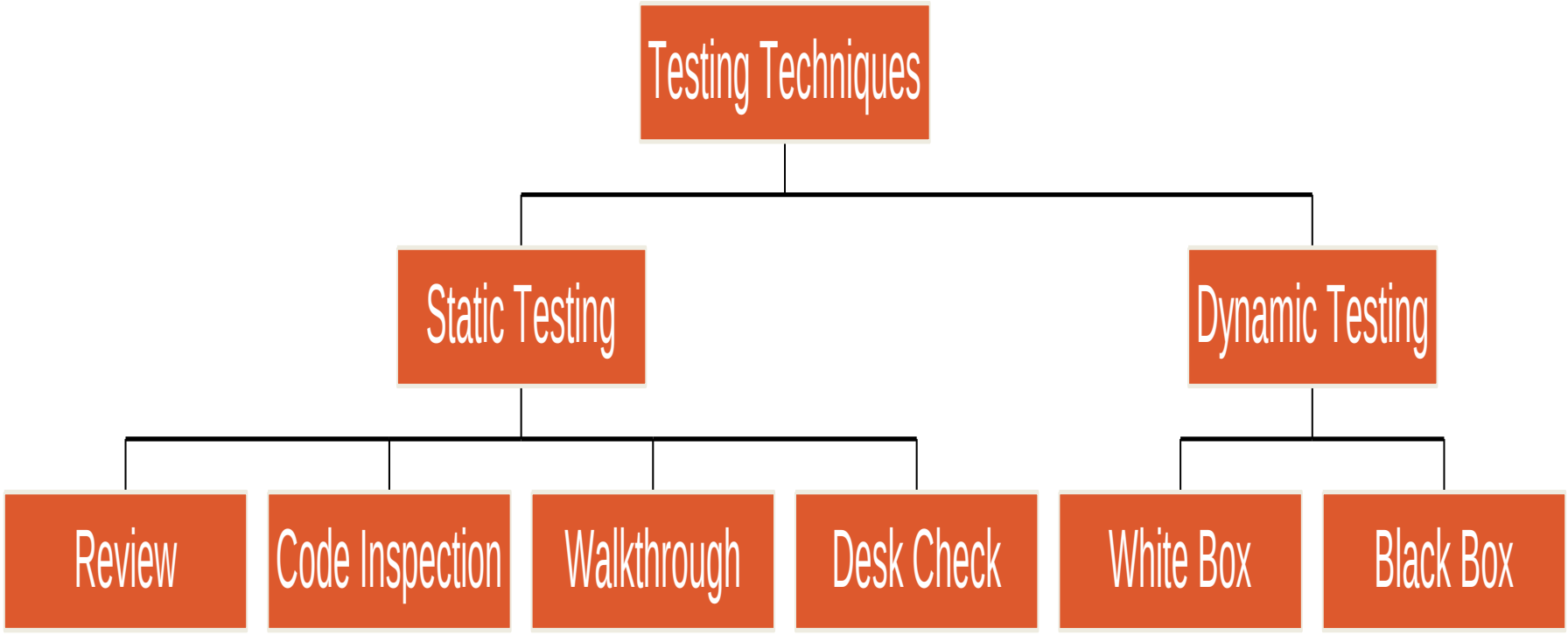
- Functional testing
- Regression testing
- Performance
- Volume
- Stress
- Load
- Security
- Usability
- Recovery
- Documentation
- Installation
- Localization
- Retesting

Types

- **Static Testing - Testing a software without execution on a computer. Involves just examination/review and evaluation**
- **Dynamic Testing - Testing software through executing it**

Types (Contd.)

Types of Testing Techniques



Black Box Test design Techniques

- A test design technique basically helps us to select a good set of tests from the total number of all possible tests for a given system.
- Using test design technique's the tester can design effective test cases that can find more defects.
- Test design techniques reduces number of tests required to test the particular functionality.

Black Box Test design Techniques

- Equivalence Partitioning
- Boundary Value Analysis
- Error Guessing
- Cause Effect Graphing
- State transition testing

Error Guessing

- **Based on experience and intuition one may add more test cases to those derived by following other methodologies**
- **It is an ad hoc approach**
- **The basis behind this approach is in general people have the knack of "smelling out" errors**

Error Guessing

- **Make a list of possible errors or error-prone situations and then develop test cases based on the list**
- **Defects' history are useful**
- **Probability that defects that have been there in the past are the kind that are going to be there in the future**
- **Some examples :**
  - Empty or null lists/strings
  - Zero occurrences
  - Blanks or null character in strings
  - Negative numbers

Error Guessing

- **Example : Suppose we have to test the login screen of an application**
- **An experienced test engineer may immediately see if the password typed in the password field can be copied to a text field which may cause a breach in the security of the application**
- **Error guessing testing for sorting subroutine situations**
  - The input list empty
  - The input list contains only one entry
  - All entries in the list have the same value
  - Already sorted input list

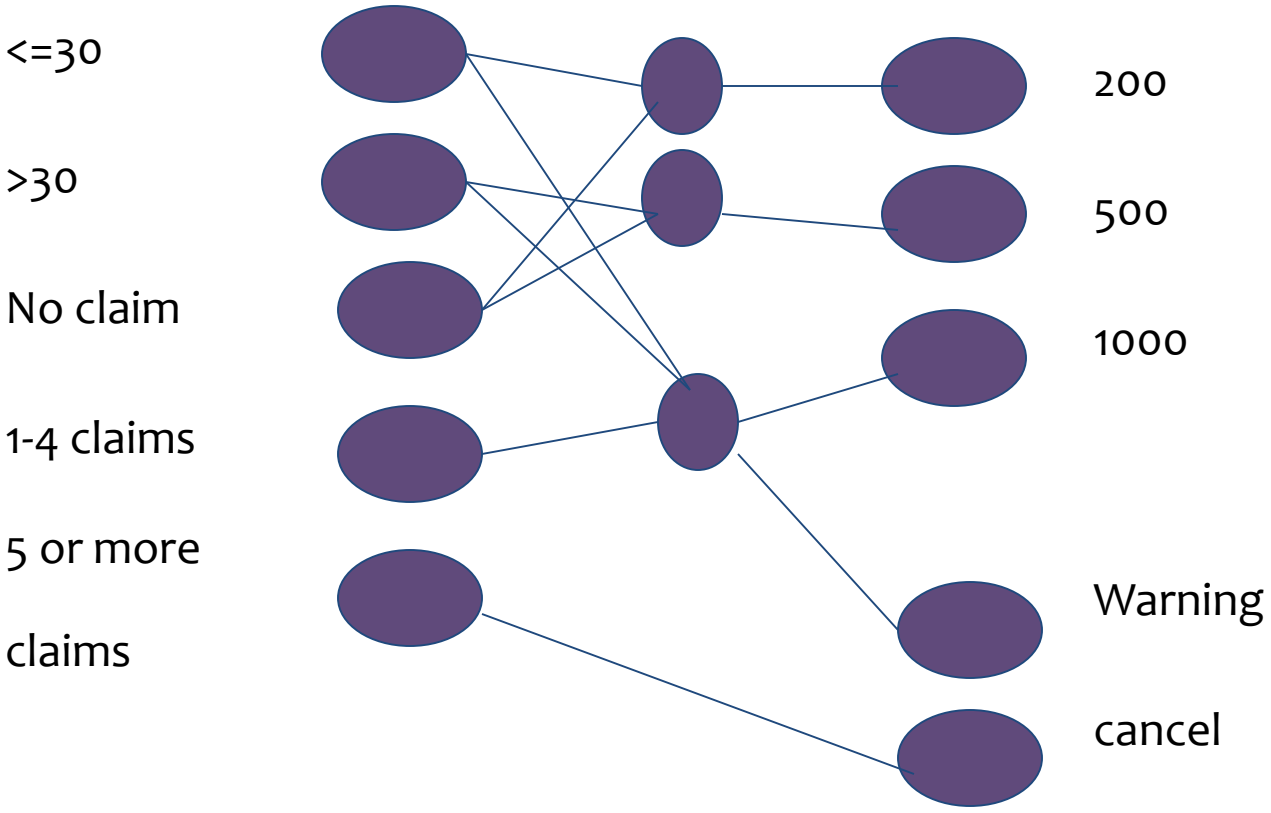
Cause Effect Graphing

- **A testing technique that aids in selecting, in a systematic way, a high-yield set of test cases that logically relates causes to effects to produce test cases**
- **It has a beneficial side effect in pointing out incompleteness and ambiguities in specifications**
- **Steps:**
  - Identify the causes and effects from the specification
  - Develop the cause effect diagram
  - Create a decision table
  - Develop test cases from the decision table

Cause Effect Graphing

- **Insurance policy renewal example**
  - An insurance agency has the following norms to provide premium to its policy holders
  - If age<=30 and no claim made premium increase will be 200 else 500
  - For any age if claims made is 1 to 4 then premium increase will be 1000
  - If one or more claims made then warning letter, if 5 or more claims made then cancel policy

Cause Effect Graphing



Cause Effect Graphing

Insurance Renewal Decision Table

No. of Claims	Insured Age	Premium Increase	Send Warning	Cancel
0	<=30 >30	200 500	No No	No No
1-4	All ages	1000	Yes	No
5 or more			No	Yes

Cause Effect Graphing

Causes	Effects
C1 age<=30 C2 age >=30 C3 No claims C4 1-5 claims C5 >5 claims	E1 Premium increase 200 E2 Premium increase 500 E3 Premium increase by 1000 E4 Warning Letter E5 Cancel Premium

State Transition Testing

- **A testing techniques that aids to validate various states when an program moves from one visible state to another.**

Menu System Example :

- **The program starts with an introductory menu. As an option is selected the program changes state and displays a new menu. Eventually it displays some information , data input screen.**
- **Each option in each menu should be tested to validate that each selection made takes us to the state we should reach next.**

State Transition Testing

- **Washing machine has different modes like soak, wash, rinse & dry**
- **Machine in these different states, exhibit different features**
  - Soak mode - clothes absorb soap water
  - Wash mode - clothes get washed with soap water
  - Rinse mode - It removes soup water from clothes
  - Dry mode - water gets removed from clothes
- **It is useful to create a state transition diagram to spot relationship between states and trace transition between states**

What is positive testing?

- **Positive testing can be performed on the system by entering the valid data as input. It checks whether an application behaves as expected with the positive input**
- **By using positive test we can check the application that does what it is supposed to do**
- **When tester test the application from positive point of mind then it is known as positive testing**
- **Testing aimed at showing software works. Also known as "test to pass"**
- **Positive testing, many times referred to as "Happy path testing" is generally the first form of testing that a tester would perform on an application**

Advantages/Limitations of positive testing

- **Advantages of positive testing**
  - Positive testing proves that a given product and project always meets the requirements and specifications
  - Positive testing is useful in the normal day to day life scenarios as it checks the expected behavior of application
- **Limitations of Positive testing:**
  - Positive tests check for only valid set of values

Example of positive testing

- **Example of positive testing**
- **Consider a scenario where you want to test an voting application which contains a simple textbox to enter age and requirement is that it should take only integers values and the value should be greater than 18. So here provide only positive integer values which are greater than 18 to check whether it is working as expected or not is the Positive Testing. Most of the applications developers are implement Positive scenarios. Testing point of view testers get less defects count around positive testing.**

Age: 19 **Enter only integer values > 18**

Positive test scenarios

- **Let's take example of Positive testing scenarios:**
  - If the requirement is saying that password text field should accepts 5 – 15 characters and only alphanumeric characters.

- **Positive Test Scenarios:**
  - Password textbox should accept 5 characters
  - Password textbox should accept upto 15 characters
  - Password textbox should accepts any value in between 5-15 chars length
  - Password textbox should accepts all numeric & alphabets values

What is negative testing?

- Negative Testing can be performed on the system by providing invalid data as input. It checks whether an application behaves as expected with the negative input. This is to test the application that does not do anything that it is not supposed to do
- When tester/User test the application from negative point of mind then it is known as negative testing
- Negative testing is the process of applying as much creativity as possible and validating the application against invalid data
- The purpose of Negative testing is to break the system and to verify the application response during unintentional inputs
- Testing aimed at showing software does not work. Also known as “test to fail”

Example of negative testing

- **Example of Negative testing:**
  - In the voting application scenario, Negative testing can be performed by testing by entering alphabets characters from A to Z or from a to z. Age text box should not accept the values or it should throw an error message for these invalid data inputs.

Age: | **Daya223** | **Enter only integer values > 18**

Advantages/Limitations of negative testing

- **Advantages of Negative Testing:**
  - Negative testing helps to improve the testing coverage of your software application under test
  - Negative testing discovers 'hidden' errors from application under test
  - Negative testing help to find more defects & improve the quality of the software application under test
  - negative testing ensures that the delivered software has no flaws
- **Limitation of Negative Testing**
  - Negative tests check for only invalid set of values

Negative test Scenarios

- **Let's take example of Negative testing scenarios:**
  - If the requirement is saying that password text field should accepts 5 – 15 characters and only alphanumeric characters.
- **Negative Test Scenarios:**
  - Password textbox should not accept less than 5 characters
  - Password textbox should not exceeds more than 15 characters
  - Password textbox should not accept special characters

What is basic test?

- **Basic tests are used to test very basic functionality of software**
- **The basic tests also verifies end to end builds**
- **Basic test are always positive tests**
- **Basic test can be smoke test or sanity test**

Example on Basic test

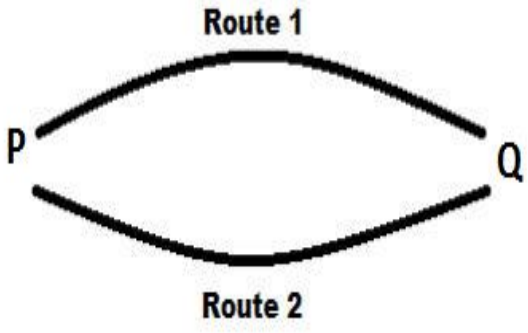
- **Customer Relationship Management (CRM) application is business philosophy towards customers. To focus on their needs and improve customer relationships, with view to maximize customer satisfaction. So, in CRM application customer creation is basic functionality that should work. So the basic test focus is on Login and then customer creation. The basic test for this CRM application is Customer login and then customer creation with mandatory fields.**

What is alternate test?

- Sometimes there maybe more than one way of performing a particular function or task with an intent to give the end user more flexibility or for general product consistency
- This is called alternate testing
- Alternate test is a kind of positive testing
- In alternate path testing the test is again performed to meet its requirements but using different route than the obvious path
- The test scenario would even consume the same kind of data to achieve the same result

Example on alternate test

- **Alternate test**

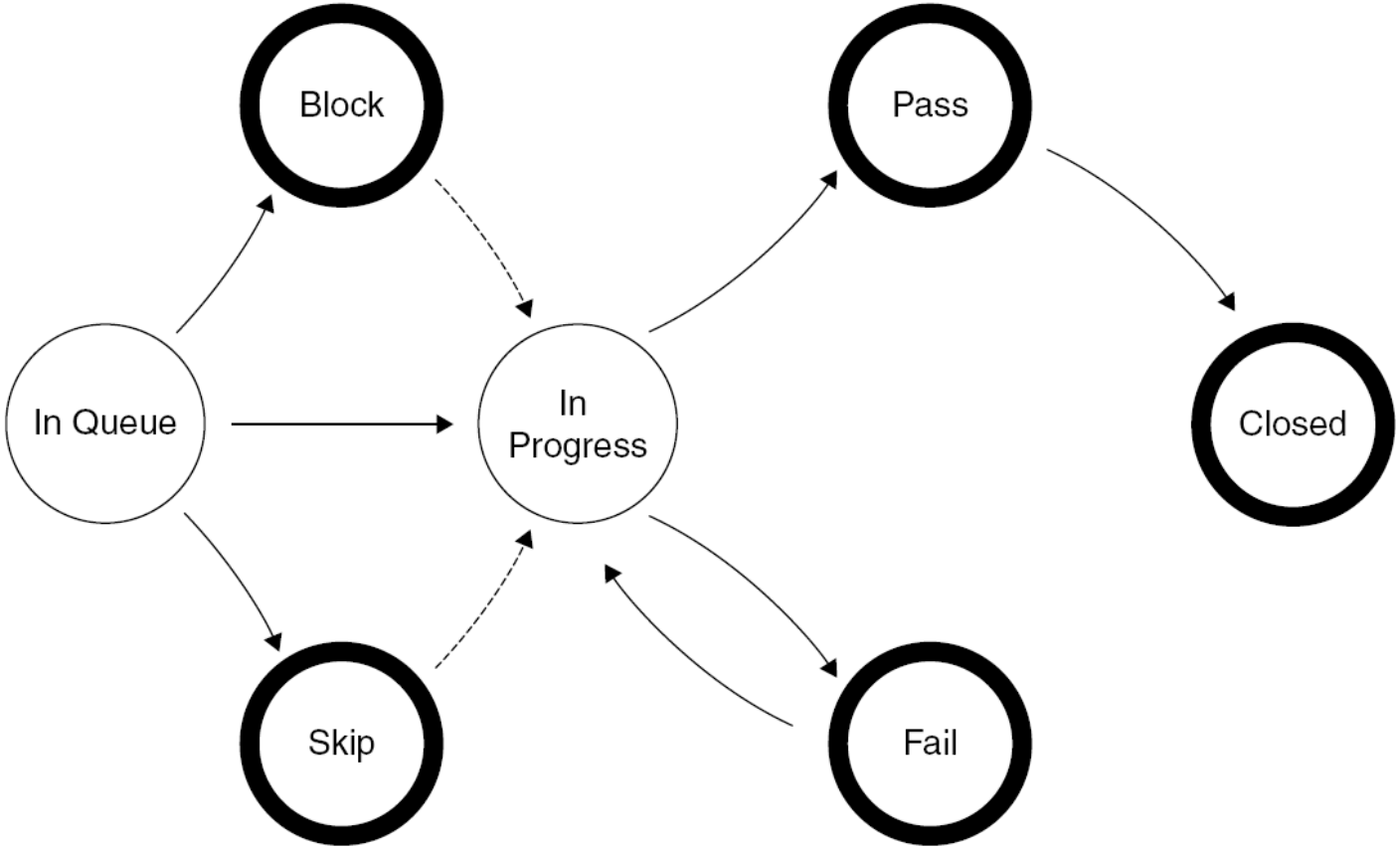


- **P is a starting point and Q is the end point. There are two ways to go from P to Q. Route 1 is the generally taken route and Route 2 is an alternative route. Therefore in such a case, happy path testing would be traversing from point P to Q using Route 1 and the alternate test would comprise taking Route 2 to go from P to Q. Observe that the result in both the cases is the same.**

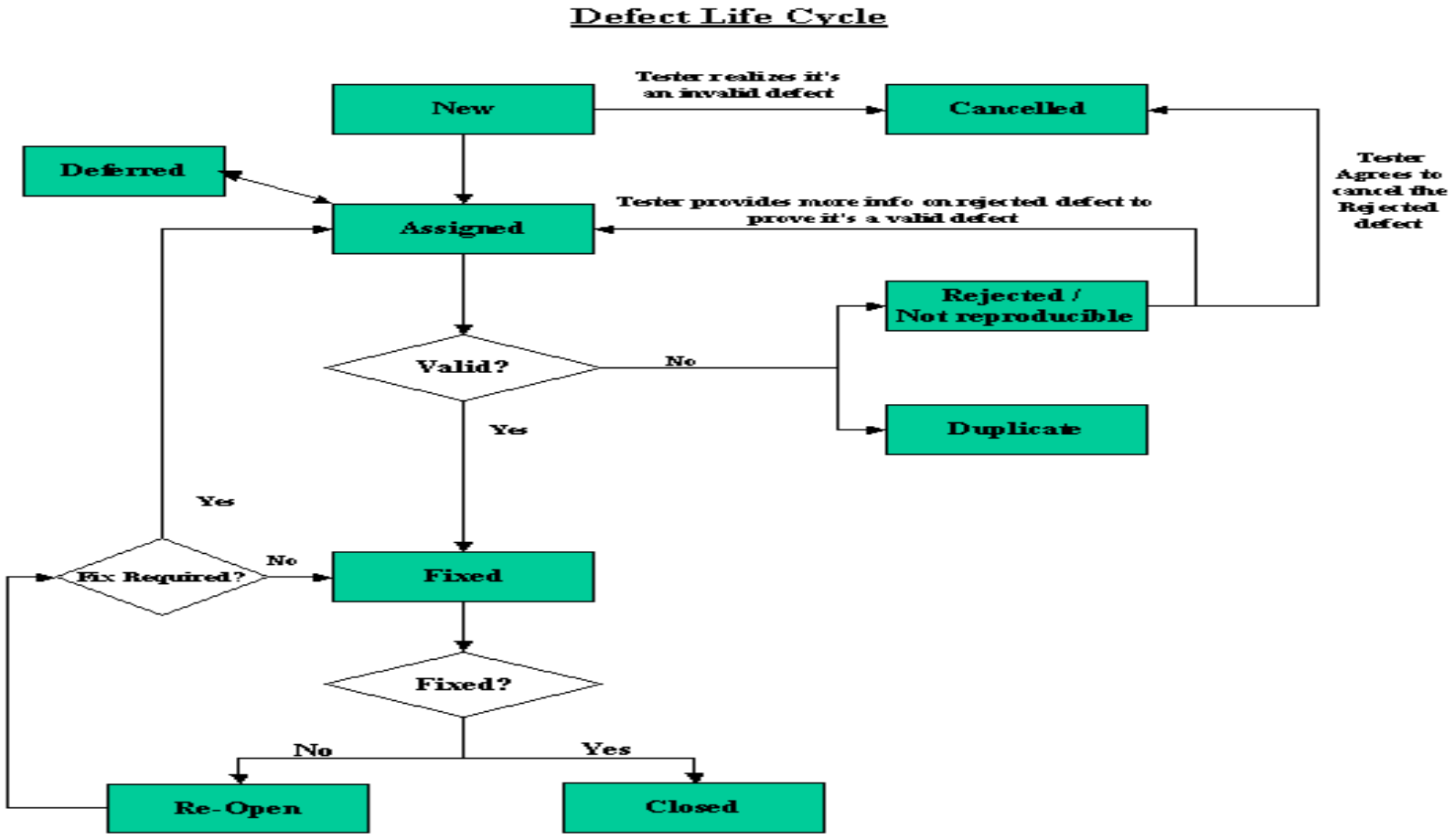
Importance of writing positive, negative, basic, alternate test while designing test cases

- **Approach of writing positive, negative, basic, alternate test are useful to design effective test cases which help to improve quality of software**
- **These approach to test case design are help to improve the test case design coverage**
- **By using these approach test cases are written for real life scenarios. It ensures real life scenarios are tested before moving software live**
- **By designing positive and negative test cases ensures that the application works as per the requirements and specifications**
- **By executing effective test cases, helps to find more defects before releasing software, so it builds confidence in system**

Test Case Lifecycle



Life-cycle workflow



Defect Severity

It is the extent to which the defect can affect the software. You can say in other words it defines the impact that a given defect has on the system.

Defect Priority

Priority defines the order in which we should resolve a defect. Should we fix it now, or we can it wait?

Scenarios related to the severity and priority

- **Few very important scenarios related to the severity and priority**
- High Priority & High Severity
- High Priority & Low Severity
- High Severity & Low Priority
- Low Priority and Low Severity

Manual Testing - Orientation Session

Thank You