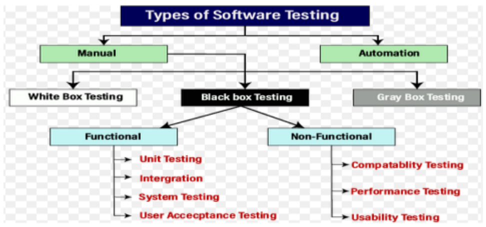
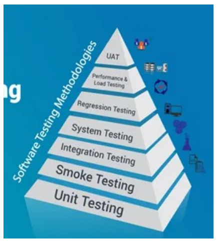
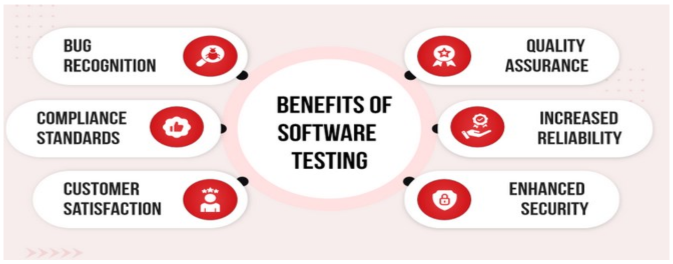




# Software Engineering And Testing

## Software Testing



## Outline

- Object Oriented Testing (OOT) Basics
- Comparison: Conventional testing and OOT
- Issues in OOT
- Issues in testing Inheritance
- Various OO Testing Techniques.
- Test Organization
- Structure of Testing Group
- Test Planning
- Detailed Test Design and Test Specification

2

## Introduction

- Object model based on data abstraction, rather than function abstraction, manages the complexity inherently.
- Reusability – faster development and high-quality s/w.
- Modeling, analysis, design, and testing of OO s/w is different than the structured approach.
- Testing an OO s/w is more challenging as most of the testing concepts lose their meaning in OOT.
- Like, in unit testing the unit is a class not module which cannot be tested with i/o behavior.
- Thus, both testing strategies and techniques are different in OO testing.

## OOT Basics

- Terminology
  - Objects
  - Class
  - Encapsulation
  - Abstraction
  - Inheritance
  - Polymorphism.
- OO Modeling and UML
  - User View
  - Structural view : Class, object diagram
  - Behavioral view: Collaboration, sequence, state chart and activity diagram.
  - Implementation view: component diagram
  - Environmental view : deployment diagram

## Differences Between Conventional Testing And Object-oriented Software Testing

| S.No. | Object-Oriented Testing | Conventional Testing |
|---|---|---|
| 1. | In object-oriented Testing, a class is considered as a unit. | In conventional testing, the module or subroutine, or procedure are considered as a unit. |
| 2. | Here, we cannot test a single operation in isolation but rather as part of a class. | Here, a single operation of a procedure can be tested. |
| 3. | It focuses on composition. | It focuses on decomposition. |
| 4. | It uses an incremental approach in the testing process. | It uses a sequential approach in the testing process. |
| 5. | This testing requires at every class level wherein each class is tested individually. | This testing is following the waterfall life cycle in its testing process. |

## Differences Between Conventional Testing And Object-oriented Software Testing

| S.No. | Object-Oriented Testing | Conventional Testing |
|---|---|---|
| 6. | This testing has a hierarchical control structure. | This testing does not have any hierarchical control structure. |
| 7. | Top-down or bottom-up integration is possible in this testing. | Here, any ordering is not possible to follow. |
| 8. | In object-oriented testing, it has unit, integration, validation, and system testing as its levels of testing. | Conventional Testing also has the same levels of testing but the approach is different. |

## Object-oriented Testing Problems

- It's not easy to test and maintain an OO software.
- **Understanding Problem**
  - Encapsulation, Information hiding are not easily understood.
  - <u>chain of invocations of member functions needs to be understood.</u>
- **State Behavior Problem**
  - Objects have state and state-dependent behavior.
  - So an operation on an object may affect its state also and so a combined effect on the s/w.

7

## Object-oriented Testing Problems

- **Dependency Problem**
  - Due to Complex relationships: *inheritance, association, aggregation, template class instantiation, class nesting, dynamic object creation, member function invocation, polymorphism, and dynamic binding* relationships.
  - For example, the inheritance relationship implies that a derived class reuses both the data members and the function members of a base class; and hence, it is dependent on the base class.

8

## Issues In OO Testing

- Following are the issues which come up while performing testing OO s/w.
- **Basic unit for testing**
  - Class is the natural unit for test case design.
  - Aggregations of classes: class clusters and application systems.
  - Class creates different test requirements: application-specific Vs general purpose classes, abstract classes and parameterized classes.
  - Testing a class instance can verify a class in isolation.
  - But when verified classes are used to create objects in application system, the entire system needs to be tested first.

## Issues In OO Testing

- **Implications of inheritance :** The inherited features need to be tested as they are in a new context after being inherited.
  - Consider multiple contexts of usage at the time of planning multiple inheritance.
- **Polymorphism :** Each possible binding of a polymorphic component requires a separate test to validate all of them.
  - But it's difficult to find all the bindings, increases the chance of errors and obstacles in reaching goals coverage.
- **White-box testing :** Conventional white-box testing cant be adopted for OOS as class is a major component to which white-box testing cant be applied.
  - They may be applicable to individual methods but not to the classes.

## Issues In OO Testing

- **Black-box testing :** In inheritance, retesting of all inherited features limits the black-box testing due to need of examination of class structure.
- **Integration strategies:** There is no obvious hierarchy of classes or methods to be followed to perform integration testing.
- So conventional methods of incremental integration testing cannot be adopted.

## Inheritance Testing

- **The issues in testing the inheritance are as follows:**
- ***Superclass modifications:*** The changed method and its all interactions must be retested.
  - Method must be retested in all the subclasses inheriting the method.
  - The ‘super’ references to the method must be retested in subclasses.
- ***Inherited Methods :*** *Extension:* inclusion of superclass features in subclass, inheriting method implementation and interface.
  - *Overriding:* the new implementation of a subclass method, with some inherited interface as that of a superclass.

## Inheritance Testing

- **The issues in testing the inheritance are as follows:**
- ***Reusing of Test suite of Superclass:*** It's not right to reuse superclass tests for  extended or overridden methods due to:
  - Incorrect initialization of superclass attributes by the subclass.
  - Missing overriding methods.
  - Direct access to superclass fields from the subclass code can create a subtle side.
  - A subclass may violate an invariant from the superclass or create an invalid state.
- ***Addition of subclass method :*** When an entirely new method is added to a specialized subclass:
  - New method must be tested with method-specific test cases.
  - Interactions between the new method and the old methods must be tested with new test cases

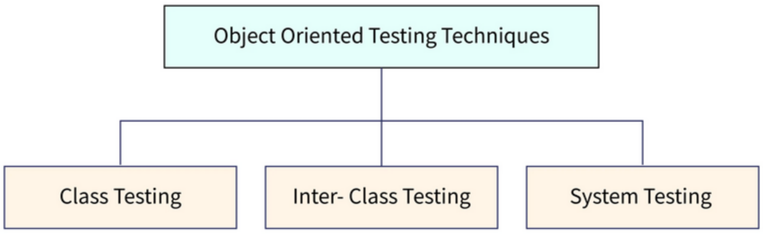
## Inheritance Testing

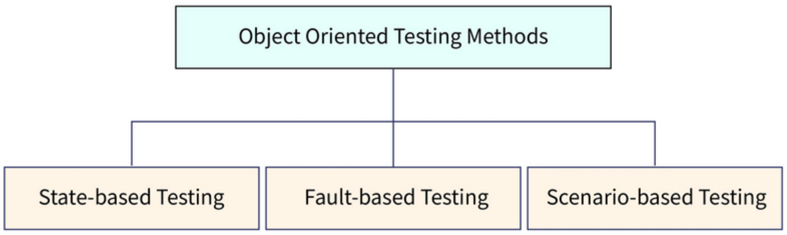
- **The issues in testing the inheritance are as follows:**
- ***Change to an abstract superclass interface :*** Abstract class is a class wherein some methods have been left unimplemented.
  - So all the subclasses must be tested with new test cases.
- ***Interaction among Methods :*** To what extent we should exercise the interactions between the methods.

## Inheritance Of Invariants Of Base Class

- An invariant defined in the base class can be extended in the subclass but do need to be tested in the subclass as the super-class data representation still complies the original invariant.
- If the invariant in the base-class is retracted than the base class must be retested with respect to the new invariant.
- Retesting can be performed by re-executing the original test cases with some additional ones to cover widened invariant.
- By clever analysis the number of test cases can be reduced from the original suite which needs to be re-executed.

## OO Testing Techniques





## OO Testing Techniques

- **Class Testing**
  1. Class testing is also known as unit testing.
  2. In class testing, every individual classes are tested for errors or bugs.
  3. Class testing ensures that the attributes of class are implemented as per the design and specifications. Also, it checks whether the interfaces and methods are error free of not.
- **Inter-Class Testing**
  1. It is also called as integration or subsystem testing.
  2. Inter class testing involves the testing of modules or sub-systems and their coordination with other modules.
- **System Testing**
  1. In system testing, the system is tested as whole and primarily functional testing techniques are used to test the system. Non-functional requirements like performance, reliability, usability and test-ability are also tested.

17

## OO Testing Methods

- **Fault-based Testing**
  1. The main focus of fault-based testing is based on consumer specifications or code or both.
  2. Test cases are created in a way to identify all possible faults and flush them all.
  3. This technique finds all the defects that include incorrect specification and interface errors.
- **Scenario-based testing:**
  1. This testing technique is useful to detect issues due to wrong specifications and improper interaction among the classes.
  2. Incorrect interactions lead to incorrect output which can cause the malfunctioning of some segments sometimes.
  3. Scenario-based testing focuses on how the end user will perform the task in a specific environment.

18

## OO Testing Methods

- **State-based Testing**
  1. The state associated with a particular object influences the methods, execution, and communication between classes. Therefore, the state plays a vital role in object-oriented testing in software testing.

## Best Unit Testing Tools

- NUnit
- JMockit
- Emma
- Quilt HTTP
- HtmlUnit
- Embunit
- SimpleTest
- ABAP Unit
- Typemock
- LDRA
- Microsoft Unit Testing Framework
- Unity Test Tools
- Cantata
- Karma
- Jasmine
- Mocha
- Parasoft
- JUnit
- TestNG
- JTest

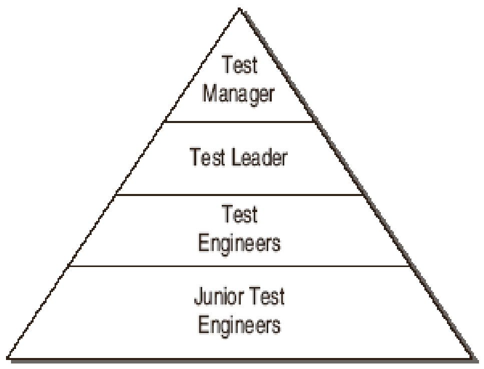
## Test Organization

- A test organization means a testing group works for better testing and high quality s/w. it is responsible for following activities:
  - Maintenance and application of test policies
  - Development and application of testing standards
  - Participation in Requirement, design, code reviews
  - Test planning
  - Test Execution
  - Test measurement
  - Test monitoring
  - Defect tracking
  - Acquisition of testing tools
  - Test reporting

21

## Structure of Testing Group

- One or two testers are not sufficient to perform testing, especially if the project is too complex and large
- Following figure shows different types of testers in a hierarchy.



22

## Test Planning

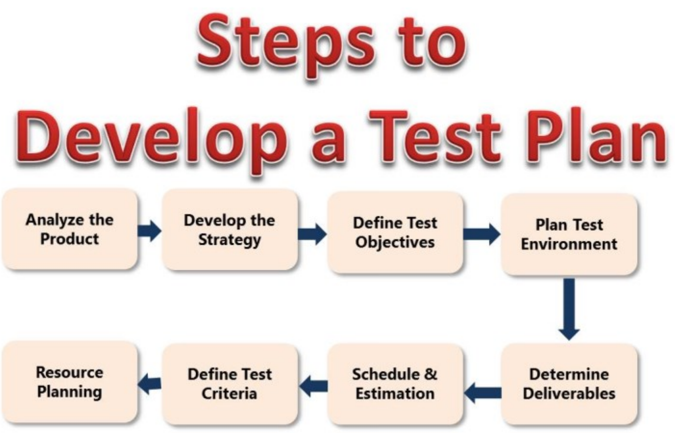
- A Test Plan is a detailed document that describes the test strategy, objectives, schedule, estimation, deliverables, and resources required to perform testing for a software product.
- Test Plan helps us determine the effort needed to validate the quality of the application under test.
- The test plan serves as a blueprint to conduct software testing activities as a defined process, which is minutely monitored and controlled by the test manager.
- As per ISTQB definition: “Test Plan is A document describing the scope, approach, resources, and schedule of intended test activities.”

## Test Planning

- A test plan identifies following to meet the goals:
  - Test items
  - Features to be tested
  - Testing tasks
  - Tools selection
  - Time and effort estimate
  - Who will do each task
  - Any risks
  - milestones

## How to write a Test Plan

- Follow the eight steps below to create a test plan as per IEEE 829

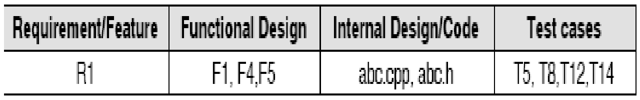


## Test Plan Components – IEEE-829

1. TestPlan Identifier
2. References
3. Introduction
4. Test Items
5. Software Risk Issues
6. Features to be Tested
7. Features not to be Tested
8. Approach
9. Item Pass/Fail Criteria
10. Suspension Criteria and Resumption
11. Requirements
12. Test Deliverables
13. Remaining Test Tasks
14. Environmental Needs
15. Staffing and Training Needs
16. Responsibilities
17. Schedule
18. Planning Risks and Contingencies
19. Approvals
20. Glossary

## Detailed Test Design and Test specifications

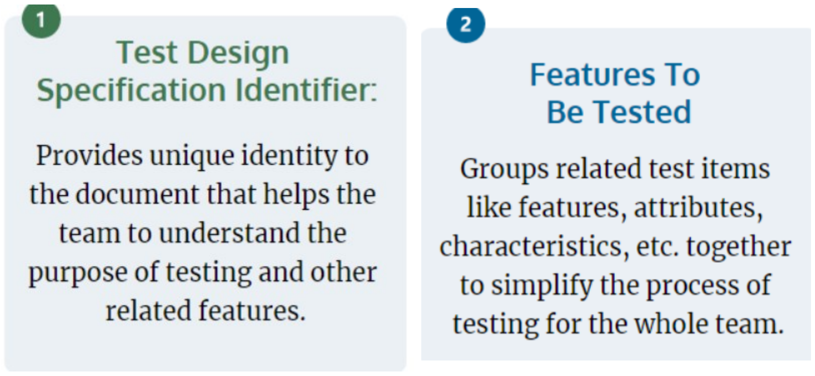
- The goal of test management is to get the test cases executed.
- Detailed test designing for each validation activity maps the requirements or features to the actual test cases to be executed.
- One way to map features to test cases is to analyze the following:
  - Requirement traceability
  - Design traceability
  - Code traceability
- Traceability Matrix



| Requirement/Feature | Functional Design | Internal Design/Code | Test cases |
|---|---|---|---|
| R1 | F1, F4,F5 | abc.cpp, abc.h | T5, T8,T12,T14 |

## Test Design Specifications

- The objective of compiling test design specifications is to identify set of features or a combination of features to be tested and to identify the group of test cases that will adequately test those features.



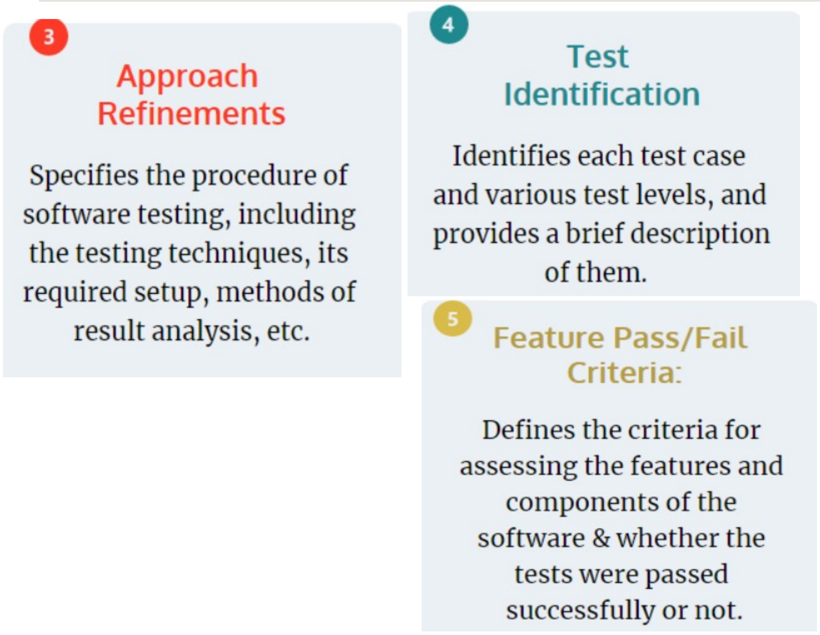
1 Test Design Specification Identifier:

Provides unique identity to the document that helps the team to understand the purpose of testing and other related features.

2 Features To Be Tested

Groups related test items like features, attributes, characteristics, etc. together to simplify the process of testing for the whole team.

## Test Design Specifications



3 **Approach Refinements**

Specifies the procedure of software testing, including the testing techniques, its required setup, methods of result analysis, etc.

4 **Test Identification**

Identifies each test case and various test levels, and provides a brief description of them.

5 **Feature Pass/Fail Criteria:**

Defines the criteria for assessing the features and components of the software & whether the tests were passed successfully or not.

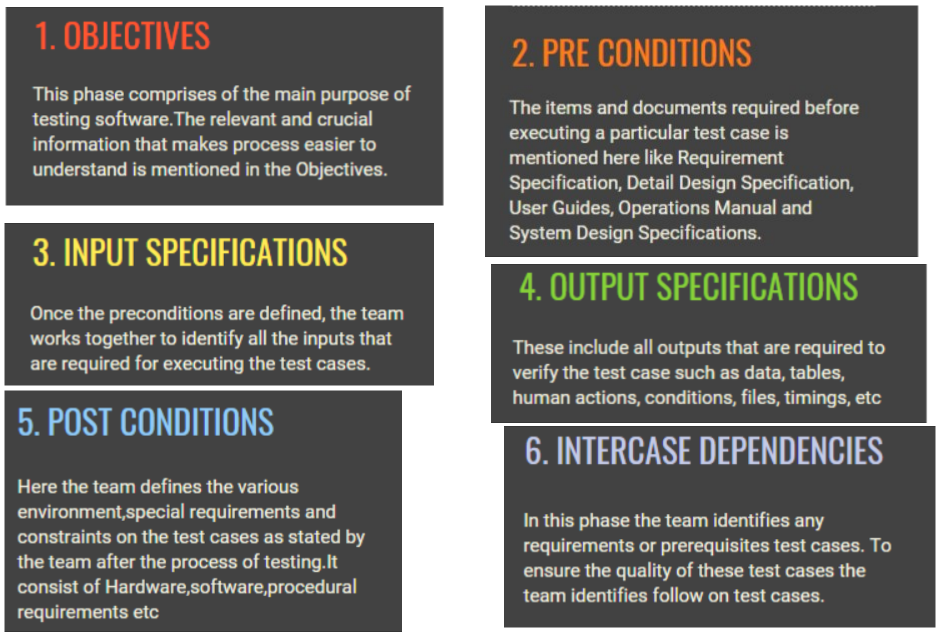
29

## Test Case Specifications

- One of the deliverables offered to the client, test case specification is a document that delivers a detailed summary of what scenarios will be tested in a software during the software testing life cycle (STLC).
- Test case specification is among those documents, whose format is set by IEEE Standard for Software & System Test Document (829-1998).

30

## Test Case Specifications



**1. OBJECTIVES**

This phase comprises of the main purpose of testing software.The relevant and crucial information that makes process easier to understand is mentioned in the Objectives.

**2. PRE CONDITIONS**

The items and documents required before executing a particular test case is mentioned here like Requirement Specification, Detail Design Specification, User Guides, Operations Manual and System Design Specifications.

**3. INPUT SPECIFICATIONS**

Once the preconditions are defined, the team works together to identify all the inputs that are required for executing the test cases.

**4. OUTPUT SPECIFICATIONS**

These include all outputs that are required to verify the test case such as data, tables, human actions, conditions, files, timings, etc

**5. POST CONDITIONS**

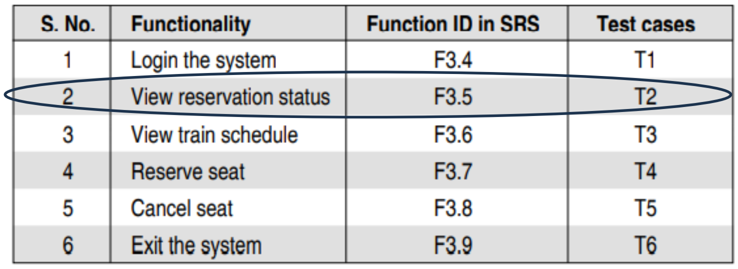
Here the team defines the various environment,special requirements and constraints on the test cases as stated by the team after the process of testing.It consist of Hardware,software,procedural requirements etc

**6. INTERCASE DEPENDENCIES**

In this phase the team identifies any requirements or prerequisites test cases. To ensure the quality of these test cases the team identifies follow on test cases.

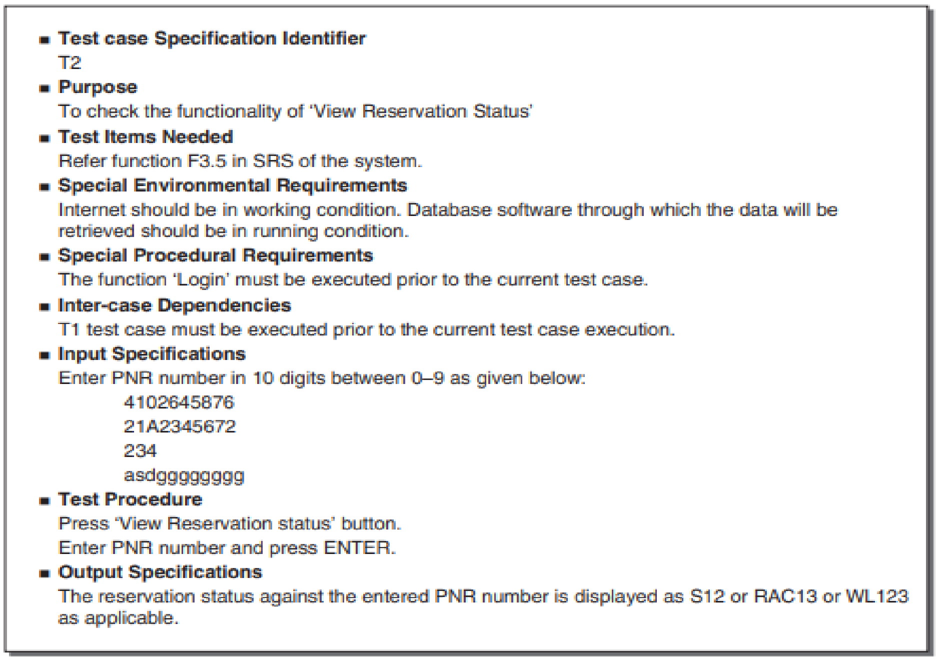
## Test Case Specifications Example

There is a system for railway reservation system. There are many functionalities in the system, as given below:



| S. No. | Functionality | Function ID in SRS | Test cases |
|---|---|---|---|
| 1 | Login the system | F3.4 | T1 |
| 2 | View reservation status | F3.5 | T2 |
| 3 | View train schedule | F3.6 | T3 |
| 4 | Reserve seat | F3.7 | T4 |
| 5 | Cancel seat | F3.8 | T5 |
| 6 | Exit the system | F3.9 | T6 |

32

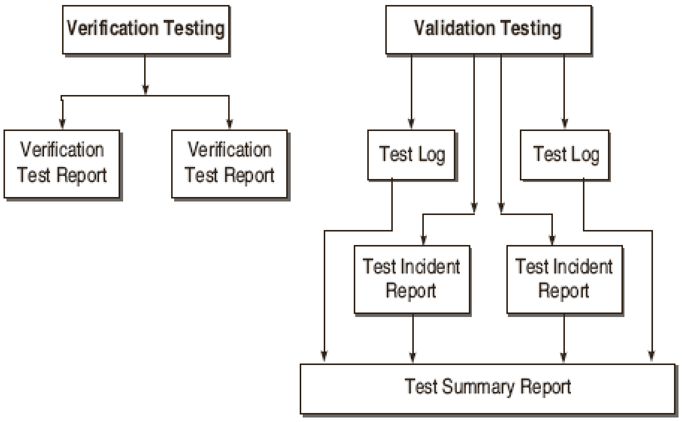


- **Test case Specification Identifier**
  T2
- **Purpose**
  To check the functionality of 'View Reservation Status'
- **Test Items Needed**
  Refer function F3.5 in SRS of the system.
- **Special Environmental Requirements**
  Internet should be in working condition. Database software through which the data will be retrieved should be in running condition.
- **Special Procedural Requirements**
  The function 'Login' must be executed prior to the current test case.
- **Inter-case Dependencies**
  T1 test case must be executed prior to the current test case execution.
- **Input Specifications**
  Enter PNR number in 10 digits between 0–9 as given below:
  4102645876
  21A2345672
  234
  asdgggggggg
- **Test Procedure**
  Press 'View Reservation status' button.
  Enter PNR number and press ENTER.
- **Output Specifications**
  The reservation status against the entered PNR number is displayed as S12 or RAC13 or WL123 as applicable.

**Figure 9.4** Test specifications

## Test Result Specifications

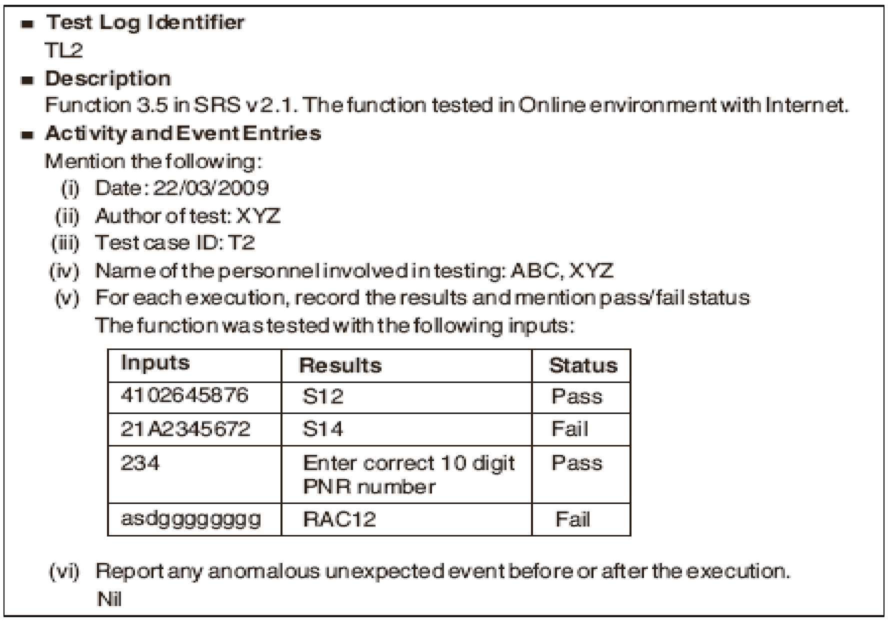
- There is a need to record and report the testing events during or after the test execution as shown below:



## Test Result Specifications

- **Test Log** : Test log is a record of testing events during test which is helpful in bug repair or regression testing.
- **The Incident Report Identifier** : A form of a bug report, which describes any unexpected event during testing which needs further investigation to resolve the bug. Also compares the expected output with the actual results.
- **The Summary Report** : Summary of all tests executed for a specific design specification at the end of testing. Can provide effort estimation for the current test. The summary will be helpful in future projects for the bugs observed in the current system.

## Test Log

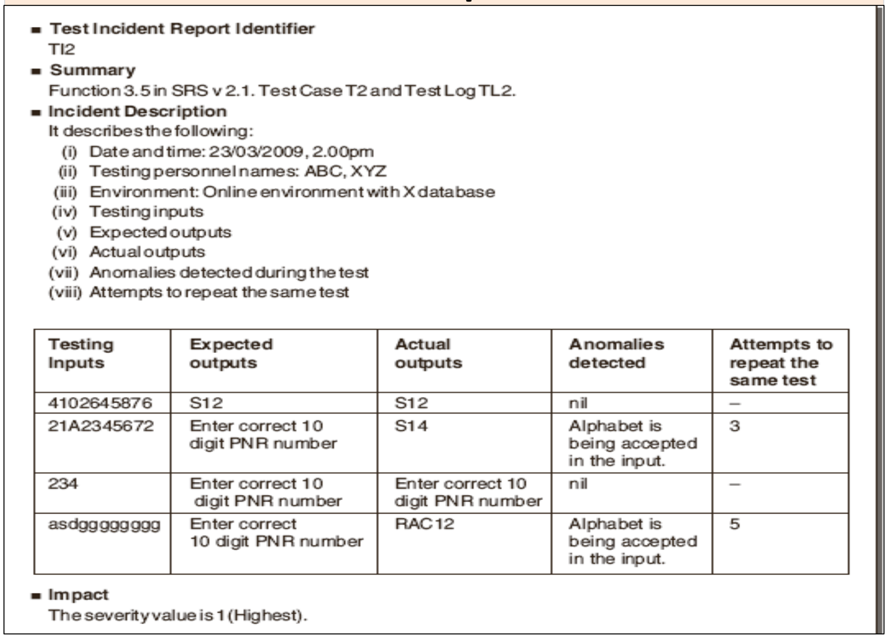


- **Test Log Identifier**
  TL2
- **Description**
  Function 3.5 in SRS v2.1. The function tested in Online environment with Internet.
- **Activity and Event Entries**
  Mention the following:
  (i) Date: 22/03/2009
  (ii) Author of test: XYZ
  (iii) Test case ID: T2
  (iv) Name of the personnel involved in testing: ABC, XYZ
  (v) For each execution, record the results and mention pass/fail status
  The function was tested with the following inputs:

| Inputs | Results | Status |
|---|---|---|
| 4102645876 | S12 | Pass |
| 21A2345672 | S14 | Fail |
| 234 | Enter correct 10 digit PNR number | Pass |
| asdgggggggg | RAC12 | Fail |

  (vi) Report any anomalous unexpected event before or after the execution.
  Nil

## Test Incident Report Identifier



- **Test Incident Report Identifier**
  TI2
- **Summary**
  Function 3.5 in SRS v 2.1. Test Case T2 and Test Log TL2.
- **Incident Description**
  It describes the following:
  (i) Date and time: 23/03/2009, 2.00pm
  (ii) Testing personnel names: ABC, XYZ
  (iii) Environment: Online environment with X database
  (iv) Testing inputs
  (v) Expected outputs
  (vi) Actual outputs
  (vii) Anomalies detected during the test
  (viii) Attempts to repeat the same test

| Testing Inputs | Expected outputs | Actual outputs | Anomalies detected | Attempts to repeat the same test |
|---|---|---|---|---|
| 4102645876 | S12 | S12 | nil | – |
| 21A2345672 | Enter correct 10 digit PNR number | S14 | Alphabet is being accepted in the input. | 3 |
| 234 | Enter correct 10 digit PNR number | Enter correct 10 digit PNR number | nil | – |
| asdgggggggg | Enter correct 10 digit PNR number | RAC12 | Alphabet is being accepted in the input. | 5 |

- **Impact**
  The severity value is 1 (Highest).