UNIT II

Black-Box Testing: What is Black-Box Testing - Why Black-Box Testing– When to do Black Box Testing – How to do Black-Box Testing-Integration Testing: Integration Testing as a Type of Testing – Integration Testing as a Phase of Testing – Scenario Testing – Defect Bash.

TEXT BOOK: “SOFTWARE TESTING Principles and Practices”-Srinivasan Desikan & Gopalswamy Ramesh, 2006, Pearson Education

Prepared by Dr.P.Radha

Black-Box Testing

- What is Black-Box Testing
- Why Black-Box Testing
- When to do Black Box Testing
- How to do Black-Box Testing

What is Black Box Testing?

- Black Box testing is done without the knowledge of the internals of the system under test.
- Black Box testing involves looking at the specifications and does not require examining the code of a program.
- Black Box testing is done from the customers view point.

Why Black Box Testing

Black Box testing helps in the overall functionality verification of the system under test.

- Black Box testing is done based on requirements.
- Black Box testing addresses the stated requirements as well as implied requirements.
- Black box testing encompasses the end user perspectives.
- Black Box testing handles valid and invalid inputs.

When to do Black Box Testing?

- Black box testing activities require involvement of the testing team from the beginning of the software project life cycle, regardless of the software development life cycle model chosen for the project.
- Testers can get involved right from the requirements gathering and analysis phase for the system under test.
- Test scenarios and test data are prepared during the test construction phase of the test life cycle, when the software is in the design phase.

HOW TO DO BLACK BOX TESTING?

Black box testing exploits specifications to generate test cases in a methodical way to avoid redundancy and to provide better coverage.

The Various techniques are

- Requirements based testing
- Positive and negative testing
- Boundary value analysis
- Decision tables
- Equivalence partitioning
- State based or Graph based testing
- Compatibility testing
- User documentation testing
- Domain testing

Integration Testing

- Integration Testing as a Type of Testing
- Integration Testing as a Phase of Testing
- Scenario Testing
- Defect Bash

Integration Testing as a Type of Testing

- Integration is both a phase and a type of testing.
- Integration testing is both a type of testing and a phase of testing.
- As integration is defined to be a set of interactions, all defined interactions among the components need to be tested.
- *Integration testing* means testing of interfaces.
- They are *internal* interfaces and *exported* or *external* interfaces.
- Integration testing type focuses on testing interfaces that are “implicit and explicit” and “internal and external.”

There are several methodologies available, to in decide the *order* for integration testing. These are as follows.

- Top-down integration
- Bottom-up integration
- Bi-directional integration
- System integration

INTEGRATION TESTING AS A PHASE OF TESTING

- Integration testing as a *phase of testing* starts from the point where two components can be tested together, to the point where all the components work together as a complete system delivering system/product functionality
- The integration testing phase focuses on finding defects which predominantly arise because of *combining* various components for testing, and should not be focused on for component or few components
- Integration testing as a type focuses on testing the interfaces.

- All testing activities that are conducted from the point where two components are integrated to the point where all system components work together, are considered a part of the integration testing phase.
- Integration testing as a phase involves different activities and different types of testing have to be done in that phase.
- This is a testing phase that should ensure completeness and coverage of testing for functionality.
- To achieve this, the focus should not only be on planned test case execution but also on unplanned testing, which is termed as “ad hoc testing.”

- The integration testing phase involves developing and executing test cases that cover multiple components and functionality
- When the functionality of different components are combined and tested together for a sequence of related operations, they are called *scenarios*.

SCENARIO TESTING

- Scenario testing is defined as a *“set of realistic user activities that are used for evaluating the product.”* It is also defined as the testing involving customer scenarios.

There are two methods to evolve scenarios.

- System Scenarios
- Use-case Scenarios/Role based Scenarios

DEFECT BASH

Defect bash is an ad hoc testing where people performing different roles in an organization test the product together at the same time.

Defect bash brings together plenty of good practices that are popular in testing industry. They are as follows.

- Enabling people "*Cross boundaries and test beyond assigned areas*"
- Bringing different people performing different roles together in the organization for testing—*"Testing isn't for testers alone"*
- Letting everyone in the organization use the product before delivery—*"Eat your own dog food"*
- Bringing fresh pairs of eyes to uncover new defects—*"Fresh eyes have less bias"*

- Bringing in people who have different levels of product understanding to test the product together randomly—“Users of software are not same”
- Let testing doesn't wait for lack of/time taken for documentation—“Does testing wait till all documentation is done?”
- Enabling people to say “system works” as well as enabling them to “break the system” — “Testing isn't to conclude the system works or doesn't work”

All the activities in the defect bash are planned activities, except for what to be tested. It involves several steps.

- *Step 1* Choosing the frequency and duration of defect bash
- *Step 2* Selecting the right product build
- *Step 3* Communicating the objective of each defect bash to everyone
- *Step 4* Setting up and monitoring the lab for defect bash
- *Step 5* Taking actions and fixing issues
- *Step 6* Optimizing the effort involved in defect bash