Software Testing Fundamentals & Methods

Software Engineering Process & Quality Management

Guest Lecture by:

Deshal Weerasinghe

BSc.IT(Curtin), MBA(USJ), CTAL-ATM

Technical Lead (QA), WSO2 Inc.

Why do we ‘Test’ Things?

Why Test Anything?

- To know whether it is ‘working’
- To know whether it ‘fits the purpose’
- To know whether ‘something is wrong’ with it
- To know ‘how it fails’, when it fails

What is Software Testing?

- Software testing consists of the **dynamic validation of** the behaviour of a program on a **finite set of test cases,** suitably **selected from the usually infinite executions** domain, against the **expected behaviour.**

(Source: SWEBOK, Chapter 5, Software Testing, 2004)

Why Software must be Tested?

- Humans develop software
- They may make errors/ mistakes
- These errors built in to software are ‘defects’
- The defects will cause the software to ‘fail’
- The failures cause damages – financial and otherwise
- **One reason to test software is to reduce (not eliminate) defects (Verification)**
- **Additionally, testing is done in order to ensure the right thing is built (Validation)**

We Test Software to Gather Information!

- “I am only a good tester when I don't care if it works or not, only that I provide the information needed to confirm or deny that it works.”

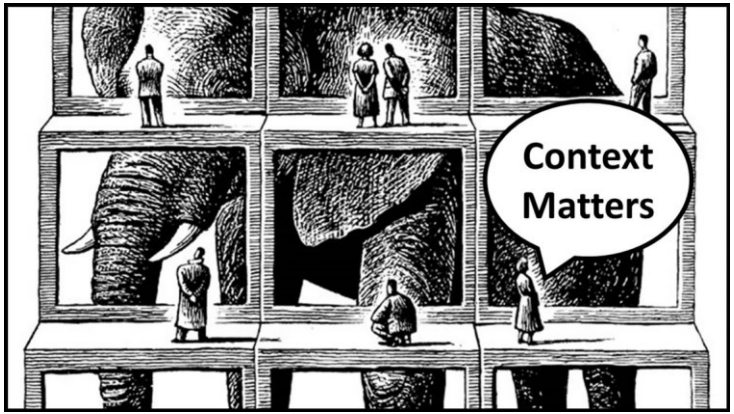
- B. Hal Metz (Test Lead Consultant, Fusion Alliance)

How Software is Tested

- Identify test context/ conditions
- Prepare test scenarios/ test cases
    - Use test design techniques
- Test in different levels of software development
- Use multiple test types
- And different test execution types to suit the task at hand
- Report test outcomes

Test Context

- Identify the type of application under test – the context
  - Eg. Bank ATM application vs. mobile email client
- Identify what features of the application to test
  - Eg. Web front end vs. API

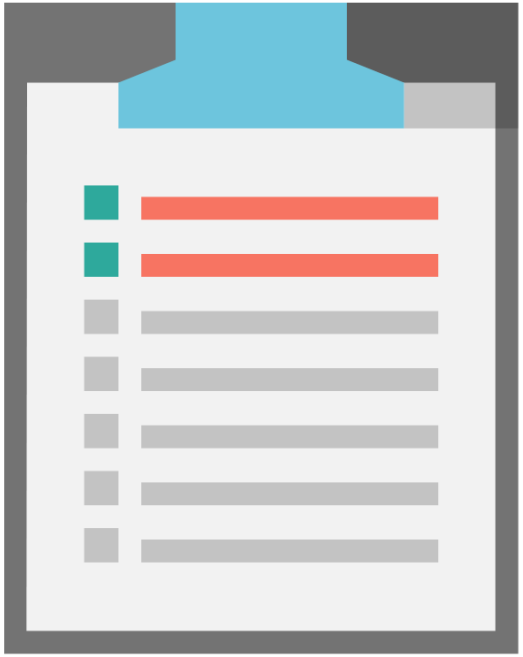


Test Scenarios & Test Cases

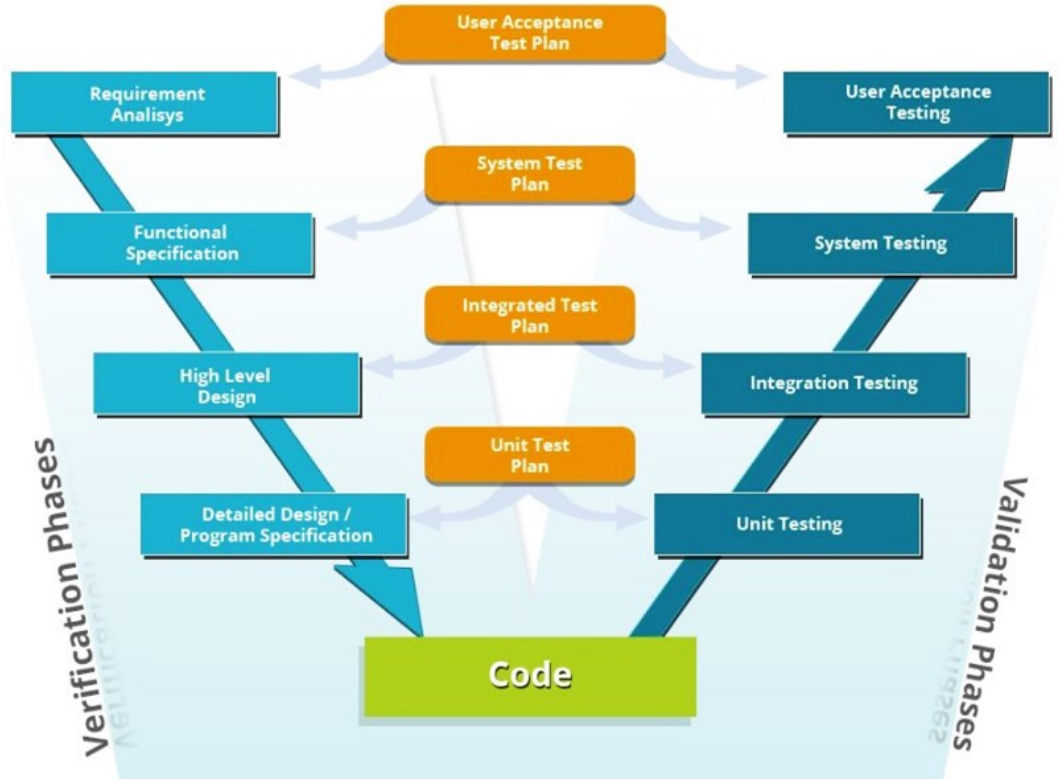
- A test scenario in general is an end-to-end flow of inter-related events
  - Eg. Log in > edit profile > change profile picture > save
- A test case defines specific inputs, operations and outputs of a selected test scenario or a part of it
- Use test design techniques to make the test scenarios efficient and effective

Components of a Typical Test Case

- Summary
- Description
- Pre-conditions
- Post-conditions (optional)
- Test data (optional)
- Test Steps
- Expected Results/ Outputs



Test Levels


User Acceptance Test Plan
Requirement Analisys
User Acceptance Testing
System Test Plan
Functional Specification
System Testing
Integrated Test Plan
High Level Design
Integration Testing
Unit Test Plan
Detailed Design / Program Specification
Unit Testing
Code
Verification Phases
Validation Phases

Test Types

- Functional Tests
- Non-functional Tests
  - Performance, load, stress
  - Usability, accessibility
- Structural
  - ‘White-box’ Testing
- Change-related Tests
  - Regression/ retesting

Test Execution Types

- Build Verification Tests (BVT)
  - Verify whether all basic functionalities of a selected application works after creating a new build with changes
- Smoke Tests
  - Verify whether a new build is stable enough to continue the rest of the testing
- Functionality Verification (’Progression’)
  - Verify one or more selected functionality after initial implementation or making changes
- Regression Test
  - Verify a functionality or related functionality to assess the impact of changes made
- Re-testing
  - Test everything once again

Test Outcomes Reporting

- Bugs/ defects
- Change requests
- Status reports
- Periodic health-checks
- Test coverage reports

How Software is Tested

- Identify test context/ conditions
- Prepare test scenarios/ test cases
    - **Use test design techniques**
- Test in different levels of software development
- Use multiple test types
- And different test execution types to suit the task at hand
- Report test outcomes

What are Test Designing Techniques?

- A test designing technique is essentially a method which ensures resulting of the optimum outcome from a test
  - Achieving maximum coverage with minimum number of tests and effort
  - Because it is practically impossible to test everything!

Test Designing Techniques

- **Dynamic Techniques** are used to test the software while executing (black-box)
  - Equivalence partitioning
  - Boundary value analysis
  - Decision tables
  - State transitions
  - Use case evaluation
- **Static Techniques** are used to test the software without execution (white-box)
  - Code analysis
  - Reviews and inspections

Black-box Techniques

- Black-box (dynamic) techniques are used to test the software during runtime, by evaluating the outputs generated for specific individual inputs or combinations of inputs

Boundary Value Analysis

- Check the behavior of the functionalities around the extreme ends of the input space
  - *Assumption: Applications behave differently on either side of a boundary in input value partitions, which causes most of the defects*
- Mostly used in combination with Equivalence Partitioning
- Limitations: Not suitable to work with Boolean variables

BVA: Example

- Age group verification – 6 to 18 years
  - Boundary Values : 6,7 – 17,18,
  - Nominal value : 12
  - Invalid values : -1, 0

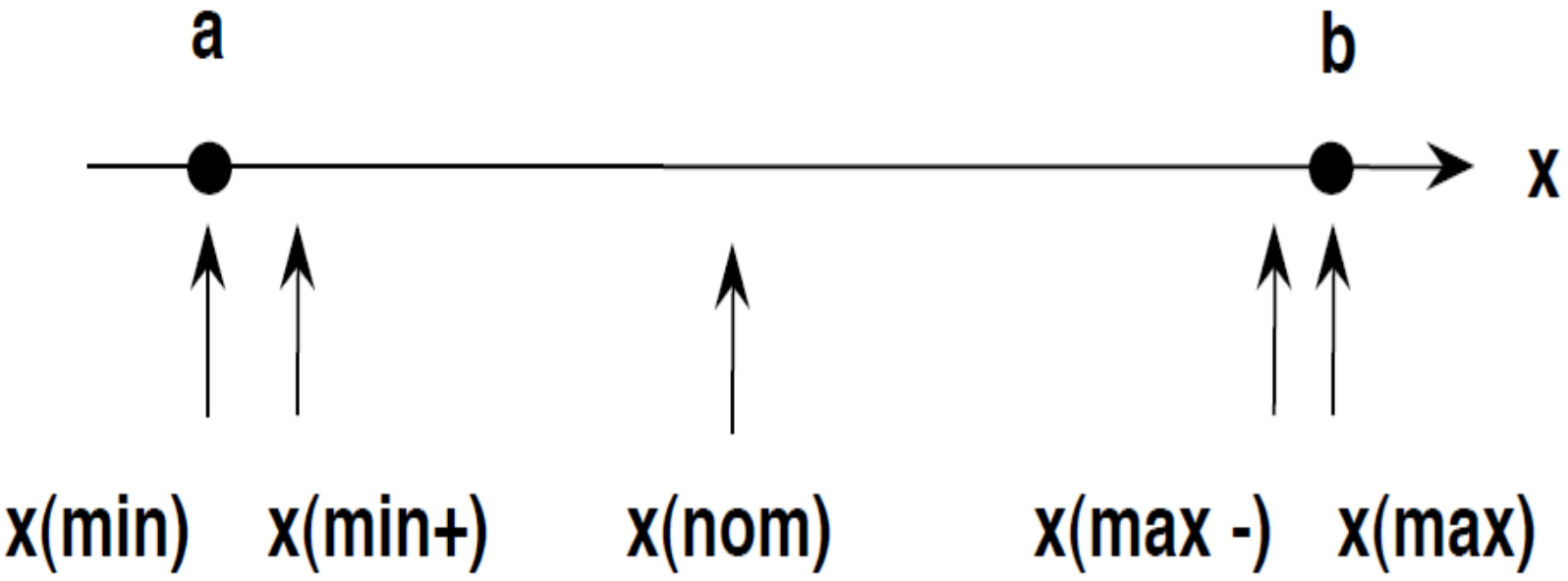
Equivalence Partitioning

- Divide the input space in to partitions
- Test each of the partitions only once
  - Assumption: All inputs within a selected partition produce the equivalent output, therefore selecting any of them does not change the output
- In most cases, EP is used in combination with BVA

EP: Example

- Age group verification – 6 to 18 years
  - Invalid partitions: Value > 18 and < 6
  - Special cases: NaN, Negative number
  - Valid partition: Value < 18 and > 6, numeric

EP & BVA



BVA & EP: Exercise

- Consider the online shopping cart of a company selling roses, there is a text box to specify the quantity
  - Only the values between 1 and 10 (inclusive) are accepted
  - For any number between 11 and 99, an error message will be displayed saying ”Max quantity allowed is 10”
  - For all other inputs, an error message will be displayed saying “Invalid quantity”
- Identify the optimum set of test inputs for this text box

Decision Tables

- Focus on logical combinations of multiple related input values
- Comes in handy when understanding and testing complex business rules
- Capture the different input combinations and their resulting outputs in a tabular format
  - AKA – Cause-effect table
- Specially important when the number of inputs in a selected functionality is high
  - Number of possible combinations = 2^n, where n = number of inputs
  - It is impossible to test all those combinations
  - Yet it is important to test a rich subset of those combinations

Decision Tables: Example

- Consider the input combinations and relevant outcomes of a user login page (username + password)

Input Fields	Combination 1	Combination 2	Combination 3	Combination 4
Username	Incorrect	Incorrect	Correct	Correct
Password	Incorrect	Correct	Incorrect	Correct
Outcomes	No log in, Show Error	No log in, Show Error	No log in, Show Error	Log in

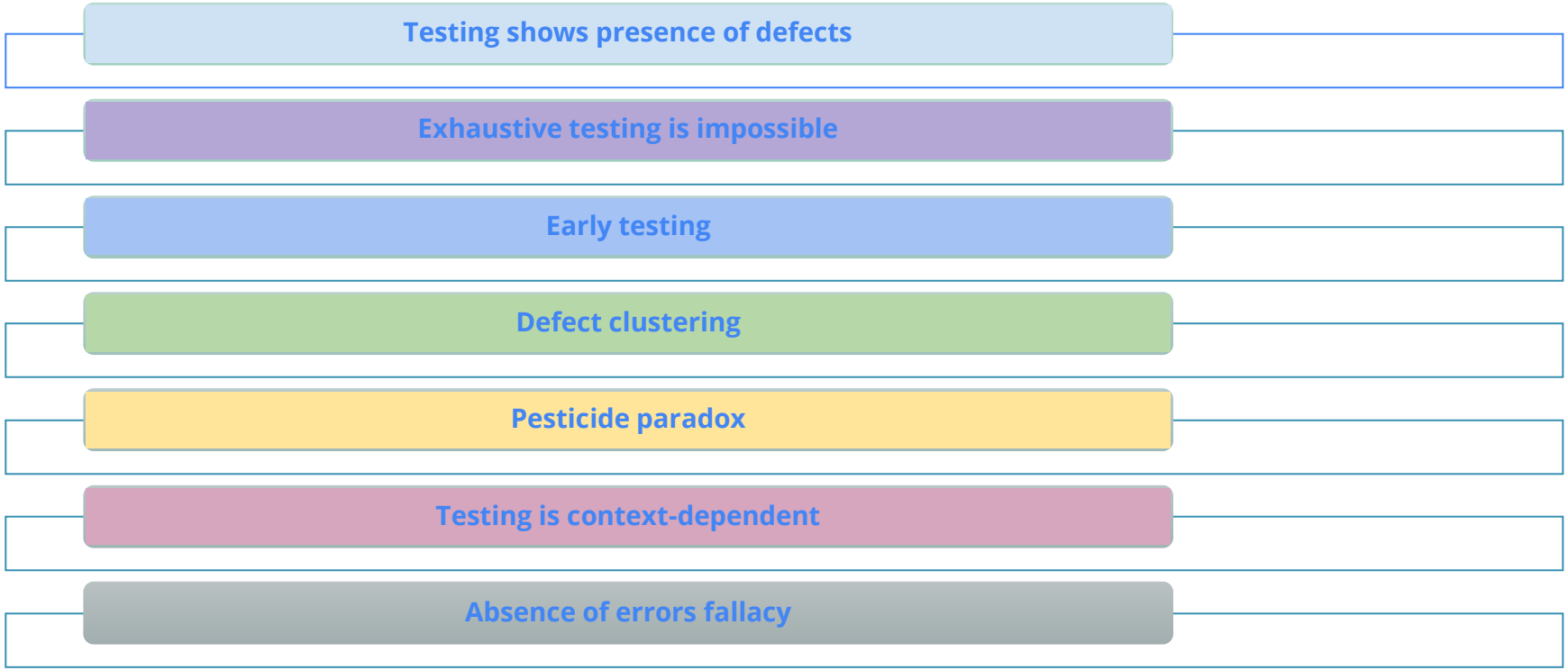
Decision Tables: Exercise

- There is a file uploader in your web application
  - Only .jpg files are accepted
  - Max file size : 1mb
  - Max resolution : 1920x1080
  - Upload the file to server if all conditions met
  - Else show an error message and stop uploading

White-box Techniques

- White-box testing techniques are used to inspect the code that has been written to deliver the functionalities
- Under that, code coverage is a measure which describes the degree of which the source code of the program has been tested
  - Statement coverage
  - Branch coverage
  - Path coverage

The Seven Principles of Testing from ISTQB



Video Links:

- Win 98 Error at press conf. https://www.youtube.com/watch?v=yeUyxjLhAxU
- Decision Tables demo 1: https://www.youtube.com/watch?v=Yl463_pRvbw
- Decision tables demo 2: https://www.youtube.com/watch?v=zHokvz4fRGY

Questions?