Unit 5: Testing

Software testing is done to ensure that a software meets its intended functionality.

Key terms in Software Testing

Testability: which describes the extent to which a software displays its faults when being tested

Quality assurance testing: is the process of ensuring the quality of the delivered software products

Validation: the assessment of a system in terms of its ability to fulfil customer's needs

Verification: checks that system developed fulfils its defined specification.

Note: It is possible that a software passes its verification checks but may not meet its validation checks

Whitebox Testing: Structure of the code is known to the tester. It is used when examining the internal operations of the system. Examples are unit testing and branch testing.

Blackbox testing: used when we want to test the overall functionality of the system. Examples are equivalence testing and use case testing.

OWASP advises that testing should be done early in the development of a system and often throughout the life cycle.

Question to ask when developing a test plan

- How many tests to run?
- What test data to use?
- Have any test cases been missed?
- When can testing stop?

Weaknesses can occur when producing software. According to Mitre reasons why weakness occur in software development include bad coding principles, data processing errors, file handling issues, and user session errors amongst others.

Examples of complexity issues include:

- Use of self-modifying code
- Excessive attack surface
- Class with excessive number of child classes
- Excessively deep nesting

Industry Software Testing Standards

OWASP Testing project provides a template that allows developers to create a testing program. The OWASP principles of testing include:

- There is no silver bullet
- The SDLC is king
- Develop the right mindset
- Understand the subject
- The devil is in the details
- Document the test results

OWASP testing techniques include: Manual Inspections & reviews, threat modelling, Source code review and Penetration testing. By using all these techniques, it is possible to test a software throughout the SDLC.

Testing Standards

ISO/IEC/IEEE/29119: consists of five parts which are intended to be applicable within any software development life cycle

Part 1: Concepts and Definitions

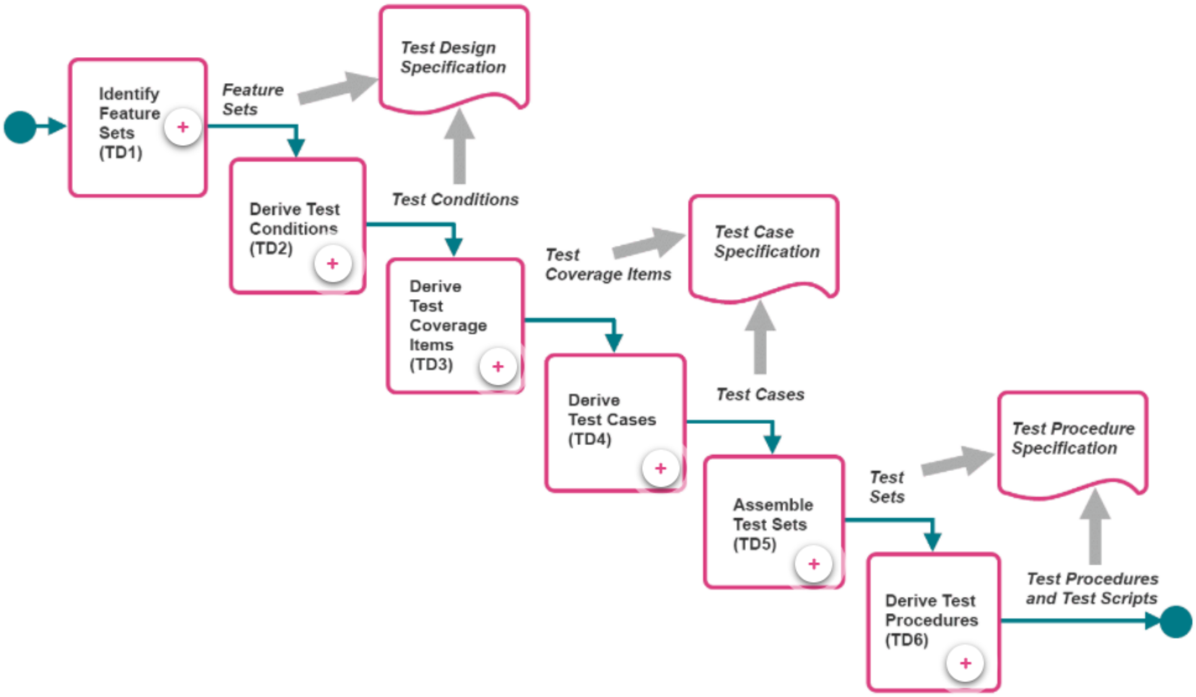
Part 2: Test Processes

Part 3: Test Documentation

Part 4: Test techniques

Part 5: Keyword Driven testing

ISO/IEC/IEEE 29119-4:2015


Identify Feature Sets (TD1)
Feature Sets
Test Design Specification
Derive Test Conditions (TD2)
Test Conditions
Derive Test Coverage Items (TD3)
Test Coverage Items
Test Case Specification
Derive Test Cases (TD4)
Test Cases
Assemble Test Sets (TD5)
Test Sets
Test Procedure Specification
Derive Test Procedures (TD6)
Test Procedures and Test Scripts

The test techniques are considered from the perspective of being specification-based, structure-based, and experience-based:

- Specification-based test design techniques consider the functional aspects of the system; these are the black-box tests. Examples include equivalence partitioning, boundary value analysis and syntax testing to name a few
- Structure-based test design techniques consider the structure of the system and are the white-box testing techniques. Examples include: branch testing, decision testing and branch condition testing amongst others.
- The Experience-based test design techniques rely on the experience of the tester and are applied to perform more random types of tests.

Linters

They are code analysis tools that help to identify problems in code styling. Most IDEs are built with linters. Python based linters are pylint, flake8, mccabe, pylama etc.

LGTM and Bandit are specifically built for security