Basic Black Box Testing Techniques

This section introduces Black Box Test Level and basic Black Box Testing Techniques.

Content:

1. Black Box Testing Level
2. Black Box Testing Techniques

1. *Black Box Testing Level*

Black Box Testing can be further refined based on the scope and details of testing. Below list the level of testing from the narrowest scope to the largest scope:

- **Functional Test**
  The foundation of Black Box Testing where test cases are narrow and focus on ensuring functionality works as per requirements or expectations. It may also be refer to Integration Test when software components are integrated to deliver a function.

- **System Test**
  System under test is tested as a whole entity without regards of modules or functions, this is to ensure the system is functioning to enable end-to-end task for the users. System Test may be functional or non-functional. Example of system tests are:
    - Performance Test
    - Stress/Load Test
    - Usability Test

- **Regression Test**
  Existing functionality of the system are tested to verify that modifications have not caused unintended effects and the system works as before, Regression test should be scenario-base and accomplish end-to-end user oriented tasks. Regression test should covers:
    - Enough of existing software functions
    - Focus on components/functions that have been changed
    - Focus on components/functions that may be affected by change

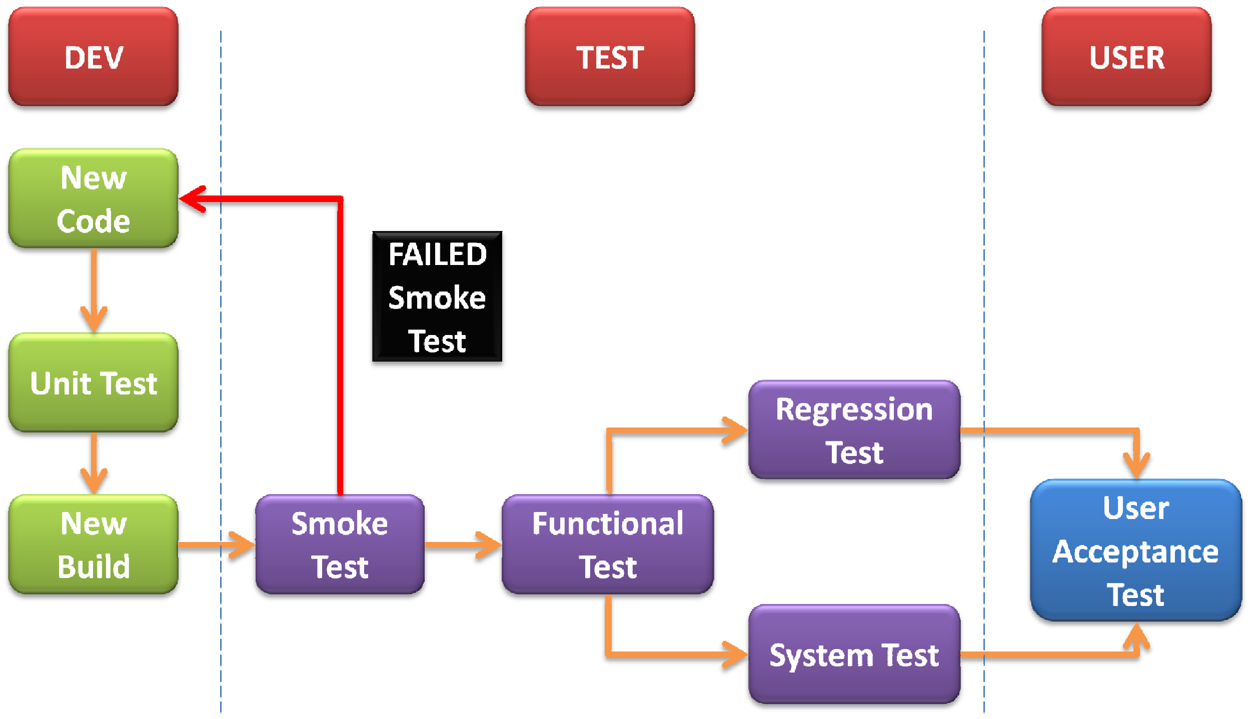
- **Smoke Test**
  Smoke test is a subset of Regression Test, with the objective is to establish that the system is stable and all **major** functionalities are present and working as expected right after a new build is created. It is preferably automated and be executed as often as possible to catch early sign of failures of a new build

- **User Acceptance Test**
  Formal testing conducted to determine if the system has satisfy its acceptance criteria and enable customer to decide if they accept the system. It is usually performed together in the present of actual users, either:
    - Execute test with users (this is much prefer)
    - Walk through system with users

The table below shows all type of tests in a glance:

Type	Specification	Scope	Opacity	Responsibility
Unit Test	Low-level design Actual code structure	Small unit of code no larger than a class	White box	developers
Integration Test	High-level design	Multiple classes	Black box	Testers
Functional	High-level design	Whole product		
System	High-level design	Whole product in representative environment		
Regression & Smoke	High-level design	Whole product		
User Acceptance	Requirements analysis	Whole product in representative environment		Customer

As point out earlier, different type of black box tests are perform as different phases, diagram below shows the different type of black box test in the software testing lifecycle.



2. *Black Box Testing Techniques*

In order to design effective test cases, here are a few techniques you can try.

- **Equivalence class**
  Dividing input domain into equal classes, the same set of data should be treated the same and produce the same results. For each equivalence class, test cases can be defined by these guidelines:
    - If input conditions specify a range of values, create one valid and one invalid value within the range
    - If input requires certain values, create one set of valid values and one set of invalid values
    - If input specify a member of a set, create one valid and one invalid (a member, a non-member)
    - If input is a Boolean, use a positive and negative values

Example:-
REQ: Application will be processed if full application fee (RM500) is paid
EQUAL CLASSES:

1. Have RM500 or more
2. Have less than RM500

- **Boundary Value Analysis (BVA)**
  Data value that corresponds to a minimum or maximum input against a specific value. BVA helps to find common error cause by >=, >, <= or <. When creating BVA test cases, consider:
    - If input conditions have a range A to B, create:
    - Immediately below A, B
    - At A, B
    - Immediately above A, B
    - If input specify a number of values, test each values and its limits

Example:-
REQ: Application will be processed if full application fee (RM500) is paid
BOUNDARY VALUES:

1. RM499
2. RM500
3. RM501

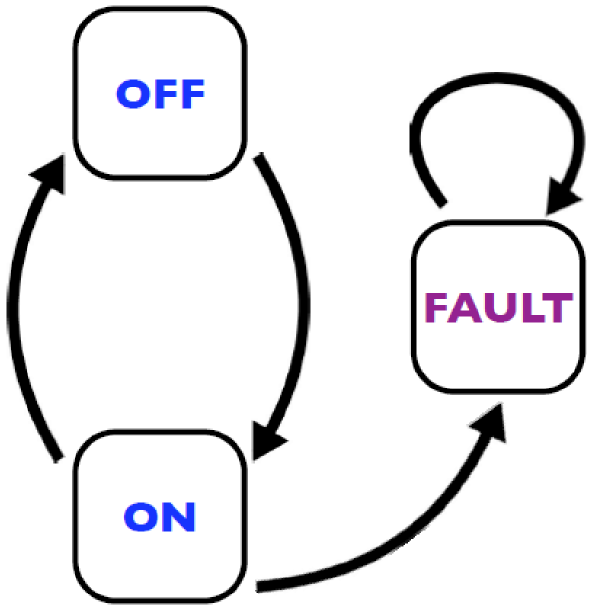
- **Decision Tables**
  Decision tables are used to record complex business rules that must be implemented and tested

Example:-
REQ: If A rent house from B, A must pay rent to B. If A doesn't have enough money, A must move out
DECISIONS:

1. Rent property? (Yes/No)
2. Enough money to pay rent? (Yes/No)
3. Stay or move out? (Stay/Move)

- **State-transition**
  Input and output of each state is verify to ensure correctness of processing

Example:-
REQ: Ensuring the On and Off switch are working correctly.



- **Negative / positive test**
  For every positive action, there is an equivalence negative action, design negative or positive test for each action. For negative action, system should response gracefully and not stop abruptly

- **Interaction Test**
  Very often, changes in one functionality will affect another functionality, such interactions need to be tested, first in isolation, the in interaction mode

Example:-
Fee collection – Finance
Hostel - Logistic
REQ: Fully paid students will have priority with accommodation arrangements)

- **Test 1**: Fee Collection – Finance
  - Verify status for students who paid their fee in full is correctly updated
- **Test 2**: Hostel – Logistic
  - Verify that student who required hostel is properly recorded and hostel is allocated
- **Test 3**: Priority in hostel when full fee is paid
  - Payment status which affect hostel allocation is correctly prioritized