Test Case Design Techniques

Thursday, December 29, 2022 11:29 AM

Testing methodologies -

White Box testing:

- In this software's internal structure, source code and design are tested to verify proper input and output functionality of the software.
- Also called as clear, transparent and glass box testing.
- Like the name White Box it is transparent to see through it and around it and test it.
- What we verify in white box,
  1.Internal structure
  2.Loopholes in the code
  3.Inputs and there respected output
  4.Functions, classes and objects
- Steps to perform testing,
  1.Understand the source code
  2.Prepare test cases and execute
- Main goal is to do code coverage testing. This has two main techniques,
  1.Statement coverage - each code statement is tested
  2.Bracnch coverage - Conditionals and loops are tested
- Can be tested by only developers and testers
- Types,
  1.Unit testing -
    In this each unit of the source code of the software is tested.
    A unit may be a function or a class.
    We test each unit separately so that final code may have very few errors.

- Advantages -
  1.Automationa testing tools are available
  2.Code becomes error free
- Disadvantages -
  1.Very complex
  2.Skilled testers required
  3.Time consuming

Black Box testing -

- In this the functionalities of a software are tested without knowing the internal structure or

source code of the software.
- Testers only considers the input and output based on test cases
- Only functional requirements and user specifications are tested in this testing
- Steps -
  1.User specifications and functional requirements are tested
  2.Test cases are made both positive and negative
  3.Expected outputs are determined for the test cases
  4.Then actual outputs are verified
- Types,
  1.Functional testing
  2.Non-functional testing
  3.Regression testing
  4.Integration testing
- No knowledge of programming structure of the software is required
- Can be tested by testers, user, anyone.

White box vs Black box -

Black Box	White Box
1.Internal structure is not known	1.Internal structure is known
2.Done by testers	2.Done by developers
3.Outer testing	3.Internal testing
4.Functional test	4.Structural test
5.No programing knowledge	5.Programing knowledge
6.Behaviour testing	6.Logic testing
7.Less time and easy	7.Time consuming and complex

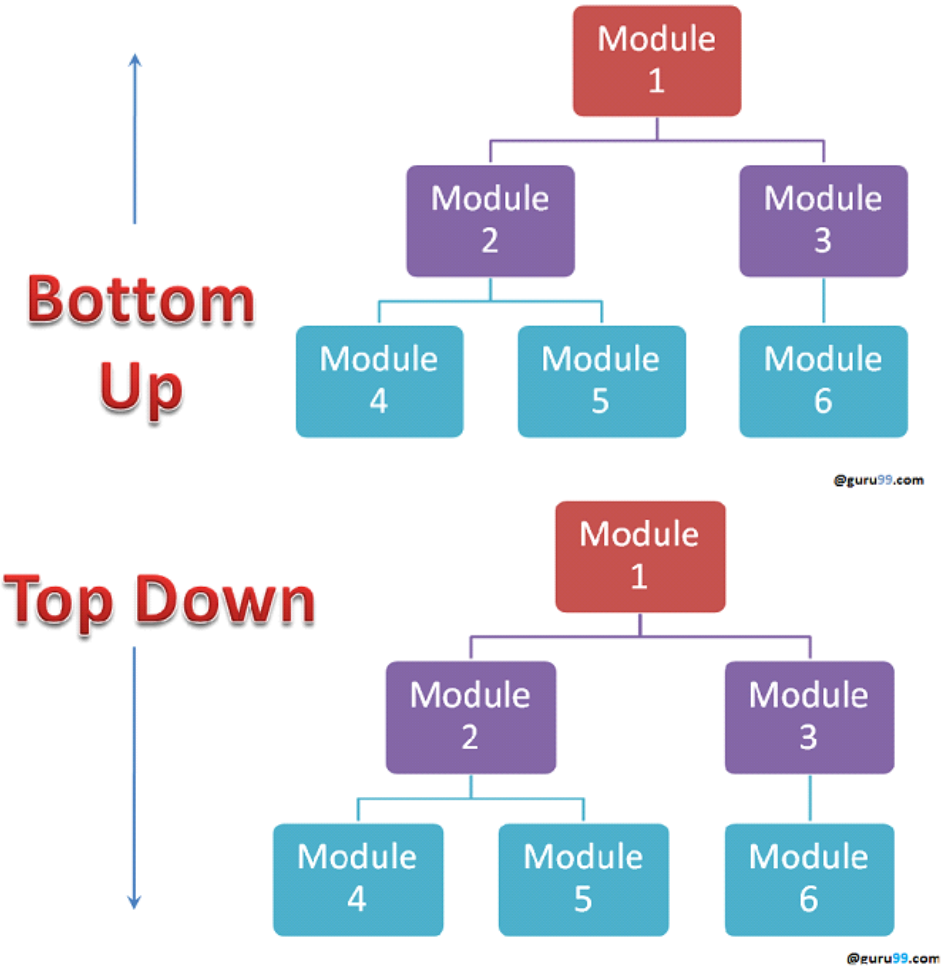
Gray box testing -

- Gray box testing is testing of the software with the partial knowledge of the internal structure of the software.
- It focuses to identify the improper implementation of code
- It combines advantages of white and black box testing
- It can check both outer structure and internal structure
- Ex. If a tester finds that some URL are not working in a web page then, tester can change the HTML code and make it functional

Integration testing -

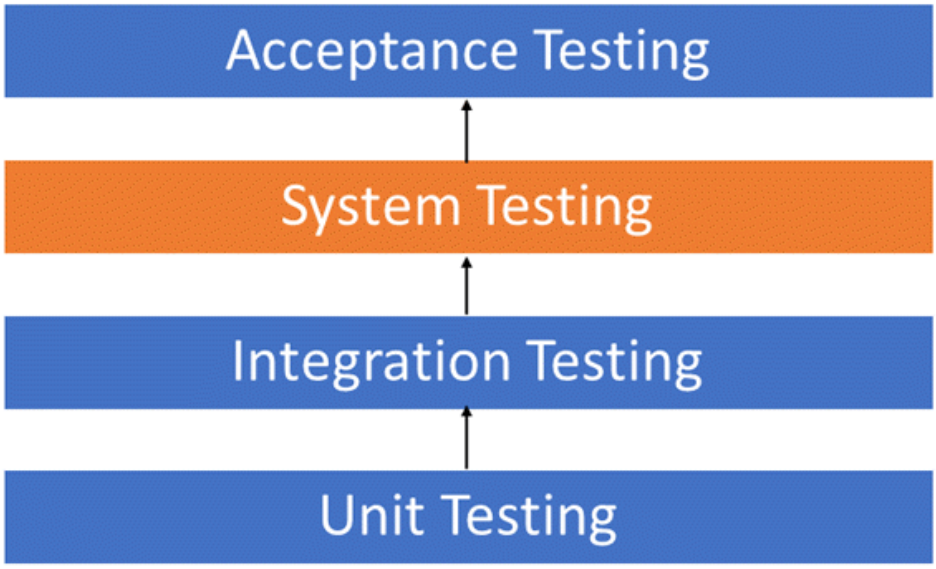
- In this the different software modules are tested to check working of the modules as a single unit.
- A software is developed in separate modules and then integrated together
- To check whether these modules work together fine we perform integration testing
- Focus is on data flow between different modules
- Ex. We have modules like login page, email page and delete mail page
- In integration testing we will check whether login page works fine with mail page and vice versa
- Types,
  1.Big bang testing
  2.Incremental testing

2.1 Top down
2.2 Bottom up
2.3 Hybrid



System testing -

- In this the complete and fully integrated software are tested as a system
- The main goal is to check end to end functionality of the software
- The software is just a component of a system
- The software is also supported by some other software and hardware to wok upon
- Testing interaction of these components with each other as a system is done in system testing.
- It is a Black Box testing method



User Acceptance testing -

- It is also called as Beta testing
- In this the software is released on test basis for the users to test it
- Users gives feedback after using the software
- These feedbacks are then used to do any modification in software
- Then another version of a software is released
- Various versions are released until a stable version is accepted by the users

Regression testing -

- It is done to check whether any code change in the software is affecting the actual functionality of the software or not.
- The executed test cases are used to again test the functionality of the software after code change
- This is imp because as requirement changes the code of the software also changes but the functionality should not change drastically
- Test cases selected as either some pre executed test cases or functionality based test cases

Retest -

- Retesting is done to check whether the failed test cases are fixed or not
- During testing when some test cases results in bug/error, these are sent back to developers to fix
- After fix these test cases are tested again
- Hence called as retest

Regression vs Retest -

Regression	Retest
1.To check functionality of software after modification in source code	1.To check whether the failed test cases are fixed or not
2.Passed test cases are executed again	2.Failed test cases are executed again

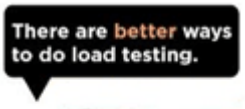
3.Does not check for defects	3.Check for defects
4.Can be done along with retest	4.Retest should be done to know any bug present
5.Not that much important	5.Very important
6.Can be automated	6.Can not be automated

Performance testing -

- It is used to test the speed, response time, resource handling, etc. of a software
- It is used to find the performance bottlenecks of a software
- The performance of a software must be tested to get its speed, scalability and stability
- Types,

1.Load testing -

It is used to test the software's working under expected load by the user
It is used to find load bottlenecks of software.



Load Testing Guidelines

2.Stess testing -

It is used to test the working of a software under extreme workloads by the user
It is used to test software under extreme load and find the breaking point

3.Endurance testing -

It is used to test whether the software can handle expected load for longer time

4.Spike testing -

It is used to test working of a software under sudden spike changes in usage by the user

5.Volume testing -

It is used to test the software under varying volumes of data in database

6.Scalability testing -

1.When software usage is scaled up then how the software functions is tested using scalability testing

2.In this we try to find the ability of the software to respond to the growing need

3.It is used to ensure that the software can handle increased users, transactions, data, etc.

4.Can be used to determine user limit of a web site

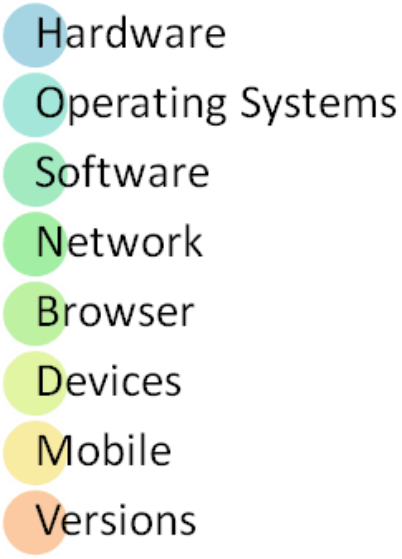
5.Can be used to test how the response to client degrades when scaling up is done

6.Also used to check server degradation after scaling up

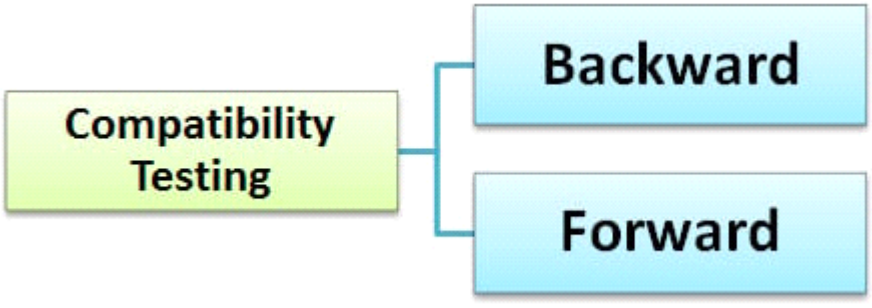
Compatibility testing -

1.Used to test whether the software is capable of running on different OS, devices, software, hardware

2.types,



3.



4.Backward compatibility testing means whether the software is compatible with its older version or software/hardware or not.

5.Forward compatibility testing means whether the software is compatible with its newer versions or new software/hardware or not

Cookie testing -

- It used to check the cookies stored on a web browser
- Cookies can be tested using various ways like,
  1.Delete cookies
  2.Change data inside cookies
  3.Does not accept cookies
  4.Test cookies on different browsers

Recovery testing -

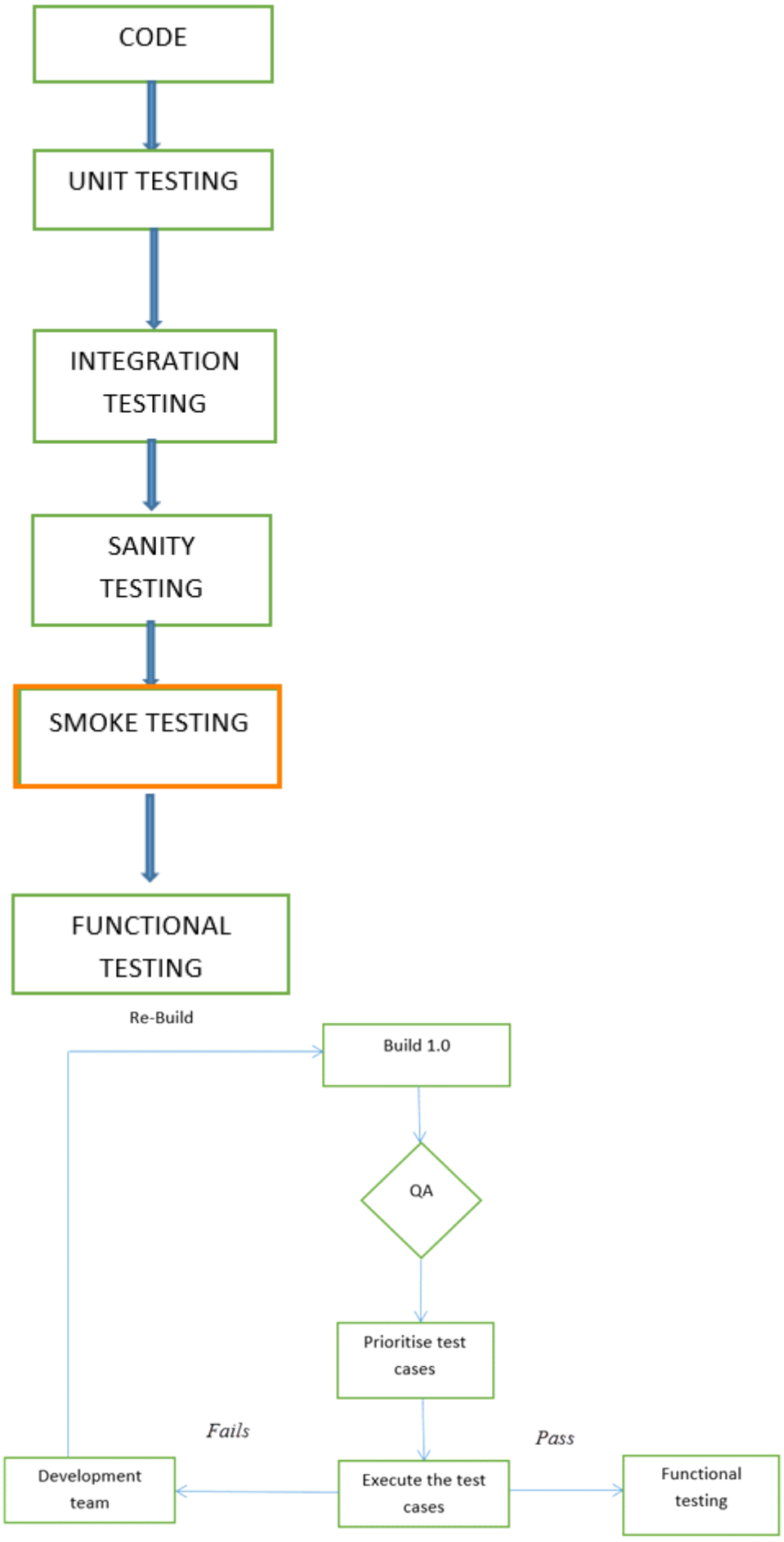
It is used to test the ability of a software to recover from a failure, data loss, connection loss, etc.

Ad hoc testing -

- It is an informal or unstructured type of testing
- This is done randomly and without any plan
- When time is limited then ad hoc testing can be done
- Any module is selected and tested without any planned test cases

Smoke testing -

- It is used to test a software's build stability
- If we wanted to test whether our software build after deploy is stable or not, we do Smoke testing
- When we add new functionality to the software and we wanted to test the stability of it then we perform smoke testing
- Ex. In Whatsapp when payment functionality is added then to check WhatsApp's stability the developers will do smoke testing

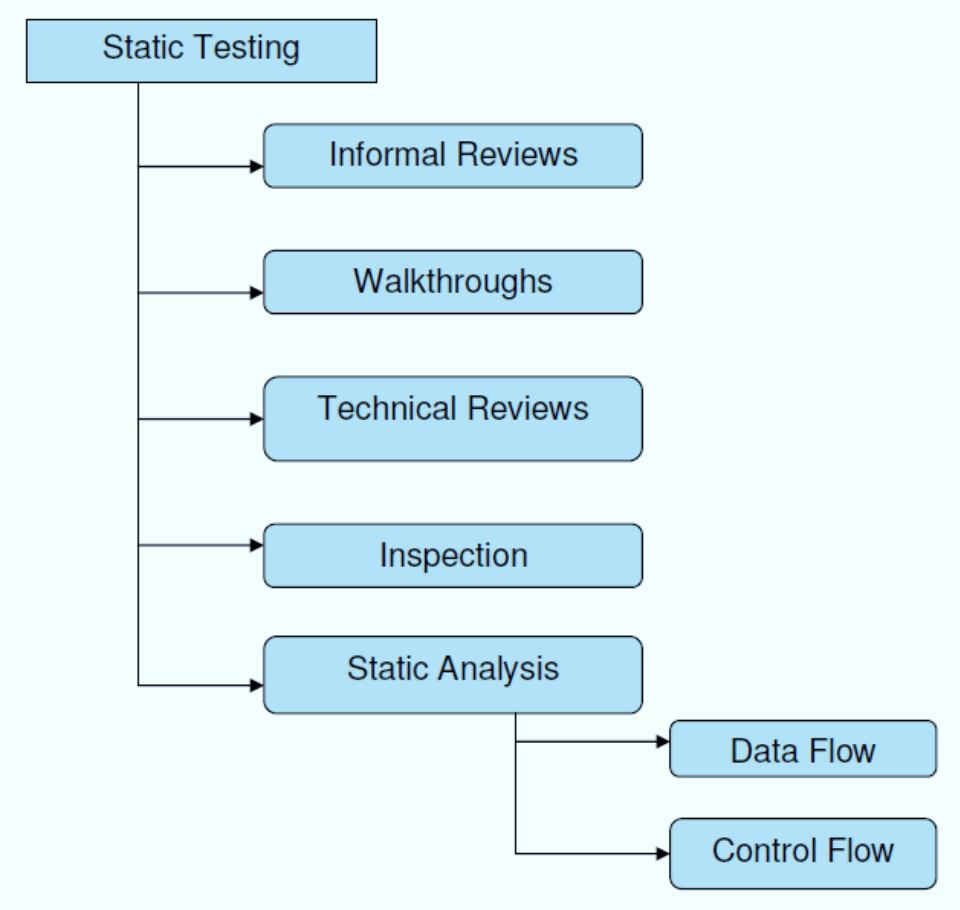

CODE
UNIT TESTING
INTEGRATION TESTING
SANITY TESTING
SMOKE TESTING
FUNCTIONAL TESTING
Re-Build
Build 1.0
QA
Prioritise test cases
Fails
Pass
Development team
Execute the test cases
Functional testing

Smoke vs Sanity testing -

Smoke	Sanity
1.To check critical functionalities of software	1.To check new functionalities and fixed bugs
2.Check stability	2.Check rationality
3.Done by developers	3.Done by testers
4.It is like general health check up	4.It is like specialized health check up

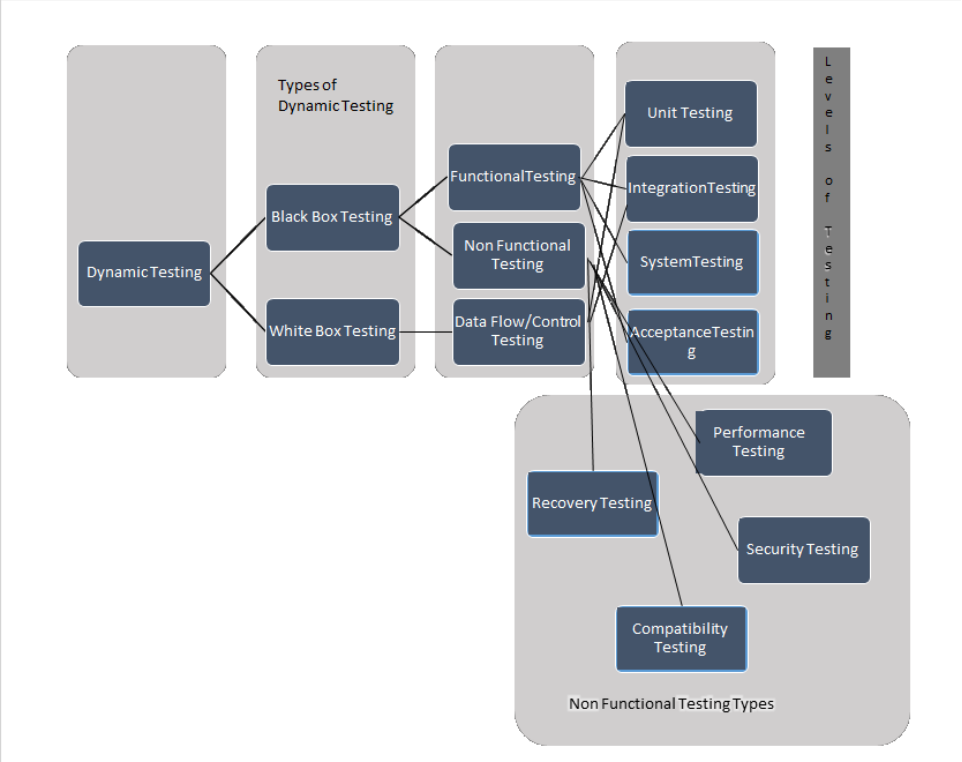
Static testing -

- In this type of testing we test a software without executing the source code
- This testing is carried to find defects at an early stage so that further defects get reduced
- Static testing can be done in two ways,
  1.Manual testing - Reviews
  2.Automated testing
- A review is like a team meeting in which defects in software are discussed and new suggestions are given
- Types of static testing are,



Dynamic testing -

- In this testing we test the dynamic behaviour of the software source code
- The source code must be executed in this testing
- In simple words, we provide some input to the software and compares actual output with expected output and check how software behaves dynamically.
- For example, there is a login page with fields username and password. The username must be only Alphanumeric.
  If we provide username as abc123 then it accepts it but if we provide abc@123 then it will display error message.
  This is called as dynamic behaviour of the software.
- Testing means verification and validation of a software.
- Verification means static testing
- Validation means dynamic testing.
- Types,



Static vs dynamic -

Static	Dynamic
Testing without executing code	Testing with executing code
Verification	Validation
Prevention of defects	Finding and fixing defects
Involves checklist	Involves test cases
Meetings are required	Meetings are not required
Before compilation	After compilation
Assessment of structure and docs	Assessment of bugs and defects

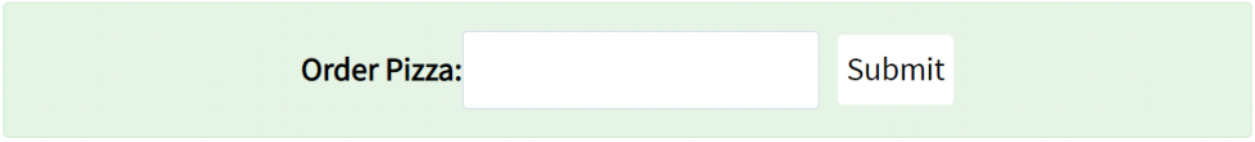
Boundary value analysis and Equivalence partitioning -

- There are 100s of test cases for a software to test
- But it is not feasible to test the software using all the test cases
- To test software using few selected test cases that can cover almost all tests we use Boundary test and Equivalence partitioning
- Boundary test means if the input to a software is given in some range of values then these range values or boundary values can be used to test the software
- Equivalence partitioning means we divide the range of input in partitions and use these partitions to test the software.
- Equivalence partitioning is done before Boundary test
- Example,

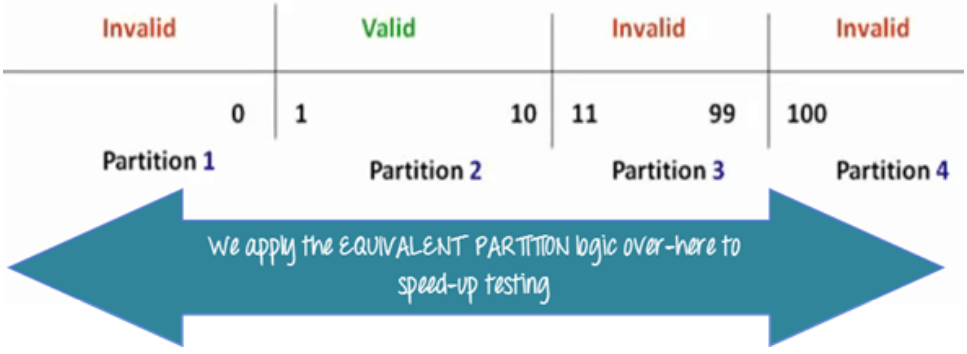
  Let's consider the behavior of Order Pizza Text Box Below

  Pizza values 1 to 10 is considered valid. A success message is shown.

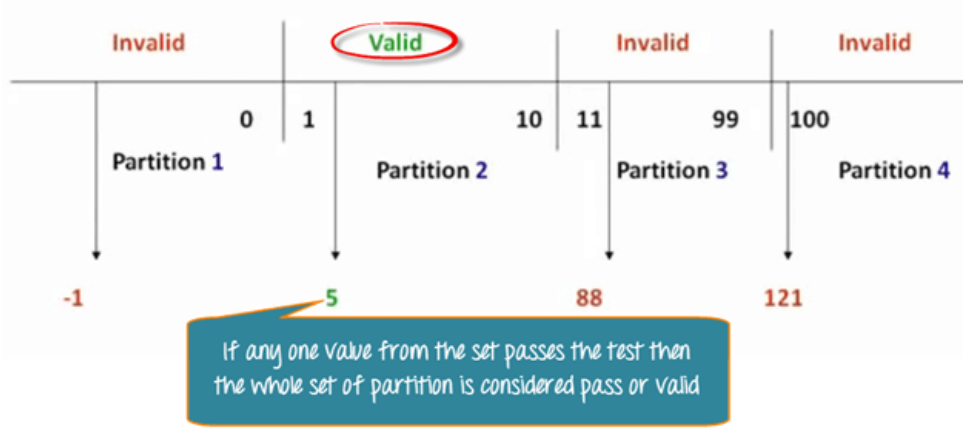
  While value 11 to 99 are considered invalid for order and an error message will appear, "Only 10 Pizza can be ordered"



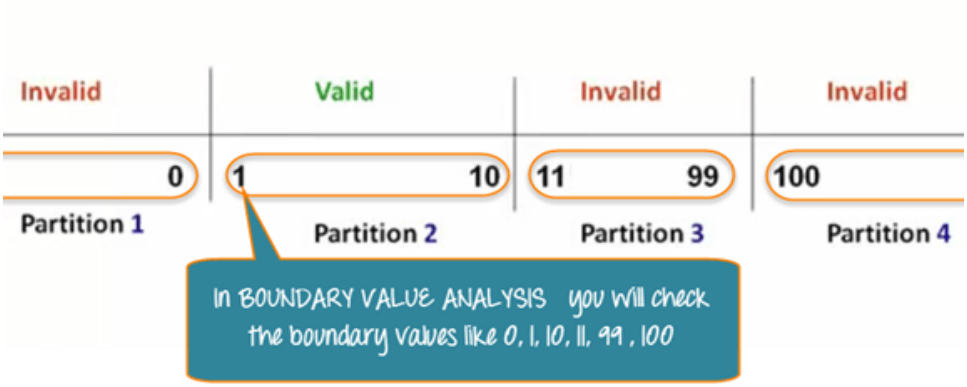
Testing all the values from 0 to 100 will not be feasible



The hypothesis behind this technique is that if one condition/value in a partition passes all others will also pass. Likewise, if one condition in a partition fails, all other conditions in that partition will fail.



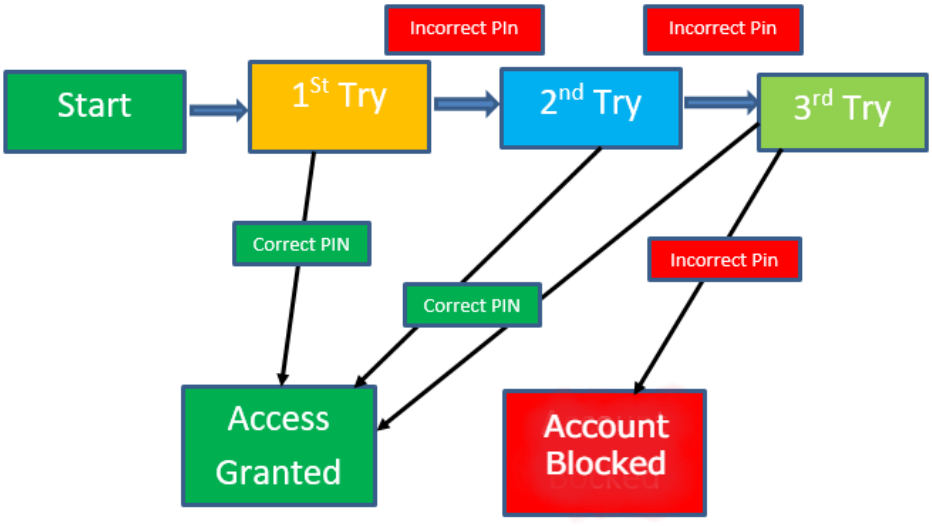
in Boundary Value Analysis, you test boundaries between equivalence partitions.



State transition testing -

- It is black box technique
- It is used to test when the change in input happens how the output changes
- Example,

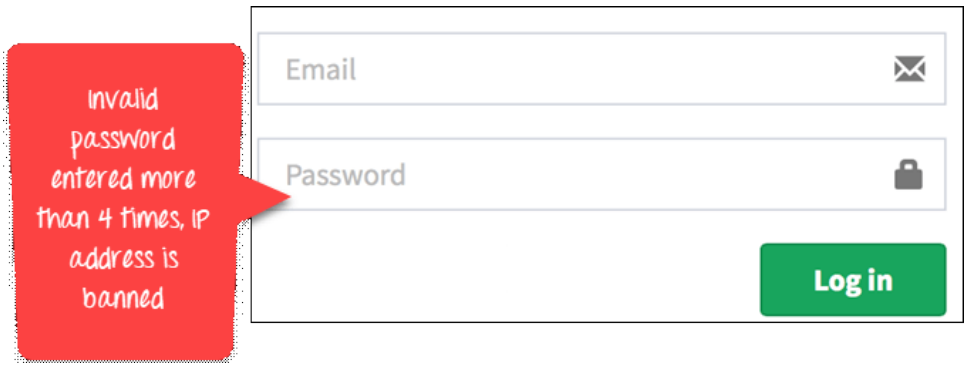
  Let's consider an ATM system function where if the user enters the invalid password three times the account will be locked.



Use case testing -

- Use case testing is used to identify those test cases that covers entire system
- A use case is description of a particular use of a software
- A use case is made of input given by user and response generated by the software

  In a use-case, an actor is represented by "A" and system by "S". We create Use for a login functionality of a Web Application as shown below



1.Consider the first step of an end to end scenario for a login functionality for our web application where the Actor enters email and password.
2.In the next step, the system will validate the password
3.Next, if the password is correct, the access will be granted
4.There can be an extension of this use case. In case password is not valid system will display a message and ask for re-try four times
5.If Password, not valid four times system will ban the IP address.