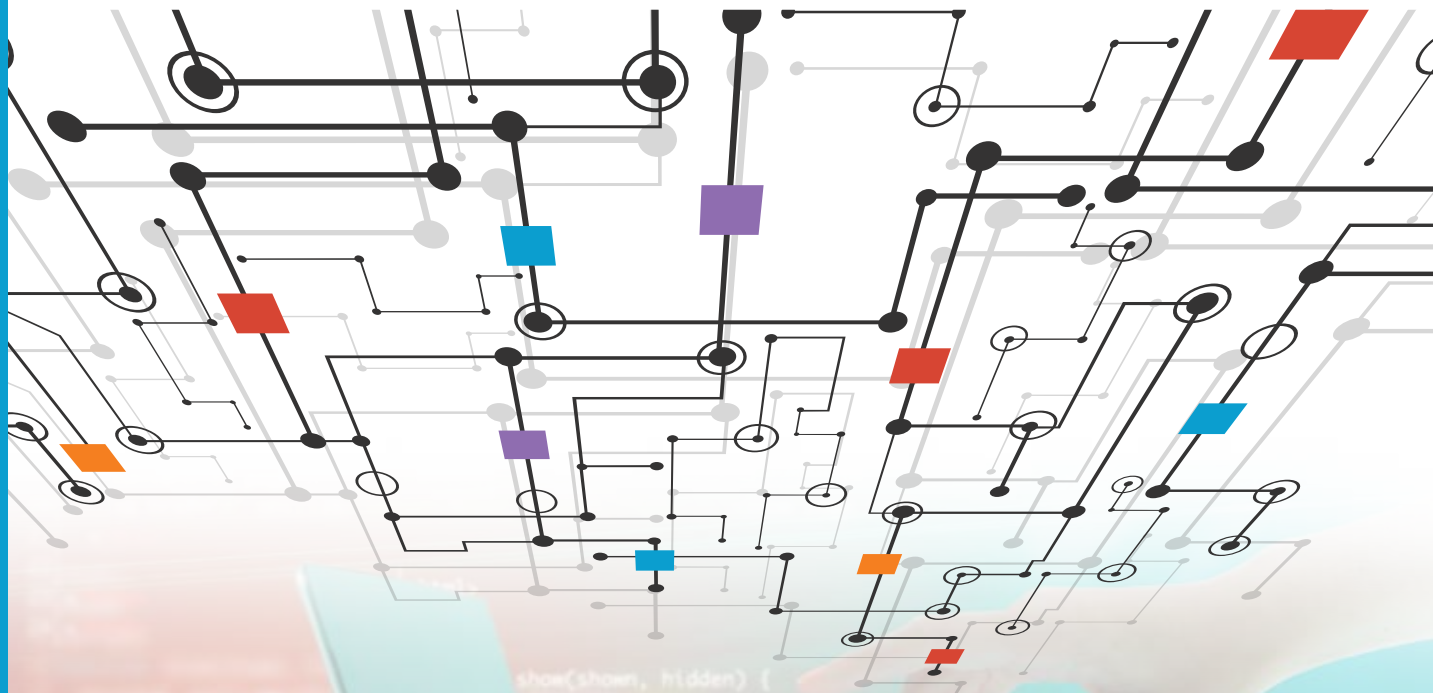
NIBM WORLD WIDE



ebook

SOFTWARE ENGINEERING DSE

SOFTWARE QUALITY ASSURANCE (SQA)

Lesson 10 – Software Quality Assurance (SQA)

What is Software Quality Assurance (SQA)?

Software Quality Assurance (SQA) is simply a way to assure quality in the software. **Software Quality Assurance is a process which works parallel to development of a software.** It focuses on improving the process of development of software so that problems can be prevented before they become a major issue. **Software Quality Assurance is a kind of an Umbrella activity that is applied throughout the software process.**

QA works out ways to **prevent possible bugs** in the process of software development. For instance, if a defect is found and fixed, there is no guaranteeing it won't pop back up. The role of QA is to identify the process that allowed the error to occur and re-engineer the system so that these **defects won't appear for the second time.**

The QA process verifies that the product will continue to function as the customer expects.

Software Quality Assurance Activities

1. Creating an SQA Management Plan:

Make a plan how to carry out the SQA throughout the project. Along with what SQA approach you are going to follow, what engineering activities will be carried out.

2. Set The Check Points:

SQA team should set checkpoints in different stages in the project to make sure the Quality of the Project being maintain. Evaluate the performance of the project on the basis of collected data on different check points. This ensures regular quality inspection and working as per the schedule.

3. Apply Software Engineering Techniques:

Software engineering techniques are the most important part to test your product. The software designer can prepare the project estimation using techniques like WBS (work breakdown structure).

4. Executing Formal Technical Reviews:

This can be carry out to evaluate the quality and design of the prototype. This activity helps in detecting errors in the early phase of SDLC and reduces rework effort in the later phases.

5. Having a Multi- Testing Strategy:

Must not rely on a single testing approach Multiple testing strategies should be performed from all angles to ensure better quality.

6. Change Controlling

Use mix of manual methods and automated tools to have a strategy for change control.

7. Measuring Change Impact:

In this phase, if any bug or error is found then the QA team will report it to the Software Engineers so that they can fix it.

8. SQA Audits

SQA audit checks that error reported by the QA team were actually performed or not by the Software Engineers.

9. Records and Reports Maintenance

The audit results, test results, change requests documentation, review reports should be kept for future reference.

10. Manage Good Relations:

Managing good relation between developer and tester is also a very important factor because they both think that they are superior to each other.

Software Quality Assurance Techniques

Auditing is the most important technique that is widely used in the Industry. Other than Auditing follow techniques can be also use.

- **Auditing:** Auditing involves inspection of the work products and its related information.
- **Reviewing**: Software is examined by both the internal and external stakeholders.
- **Code Inspection:** It is the most formal kind of review that does static testing to find bugs and avoid defect growth in the later stages. It is done by a trained mediator/peer. Not the same developer who develop the Code.
- **Design Inspection:** Focus on Functional and Interface specifications, Requirement traceability, Logic, Performance, Error handling and recovery.
- **Functional Testing:** Focuses on testing the system specifications or features.
- **Walkthroughs:** In Software walkthrough or code walkthrough development team to go through the product and raise queries, suggest alternatives or any other issues.
- **Path Testing:** It is a whit box technique where the complete branch coverage is ensured by executing each independent path at least once.
- **Stress Testing:** This type of testing is done to check whether the system can function under heavy load.
- **Six Sigma:** Six Sigma is a quality assurance approach that aims at nearly perfect products or services. (The software is 99.76 % defect free)

Benefits of Software Quality Assurance

- SQA produce high quality software.
- High quality application saves time and cost.
- No need maintenance for long time.
- High quality commercial software increase the market demand.
- Improving the process of creating software.

Software Testing

It is the process of executing a system in order to find bugs (errors) in the Software to get that fixed. Testing is helps to runs the software the way it is expected and designed for.

Software testing is a process of verifying and validating that the software is,

- Error Free
- Meets the technical requirements
- Meets the user requirements effectively and efficiently
- Handling all the exceptional and boundary cases.

Figure 11.0.1 Difference between Verification and Validation

Difference between Quality Assurance and Testing

QA VS QC VS TESTING

Quality Assurance applied throughout the software process to prevent possible bugs in the process of software development.	**Quality Control** detects bugs by inspecting and testing the product. Validating that the product meets those requirements.	**Testing** is a subset of Quality Control. It is the process of executing a system in order to find bugs (errors) in the Software to get that fixed.
Quality Assurance is Process Oriented and Proactive	Quality Control is Product Oriented and Reactive	

Figure 11.0.2 QA, QC and Testing

Software Quality Assurance	Software Testing
Ensure that Software is being developing according to the Specification	The Process of executing the system in order to find errors
Assure the quality and meeting the requirements	Software inspection and bug finding
Process Oriented	Product Oriented
Preventive	**Corrective**

Table 11.0.1 Software Quality Assurance vs Testing

Software Testing Life Cycle

Software Testing Life Cycle is a process which has specific steps to ensure that the quality of the Software have been met. Different Organizations may have different phases and following are the common steps to be follow.

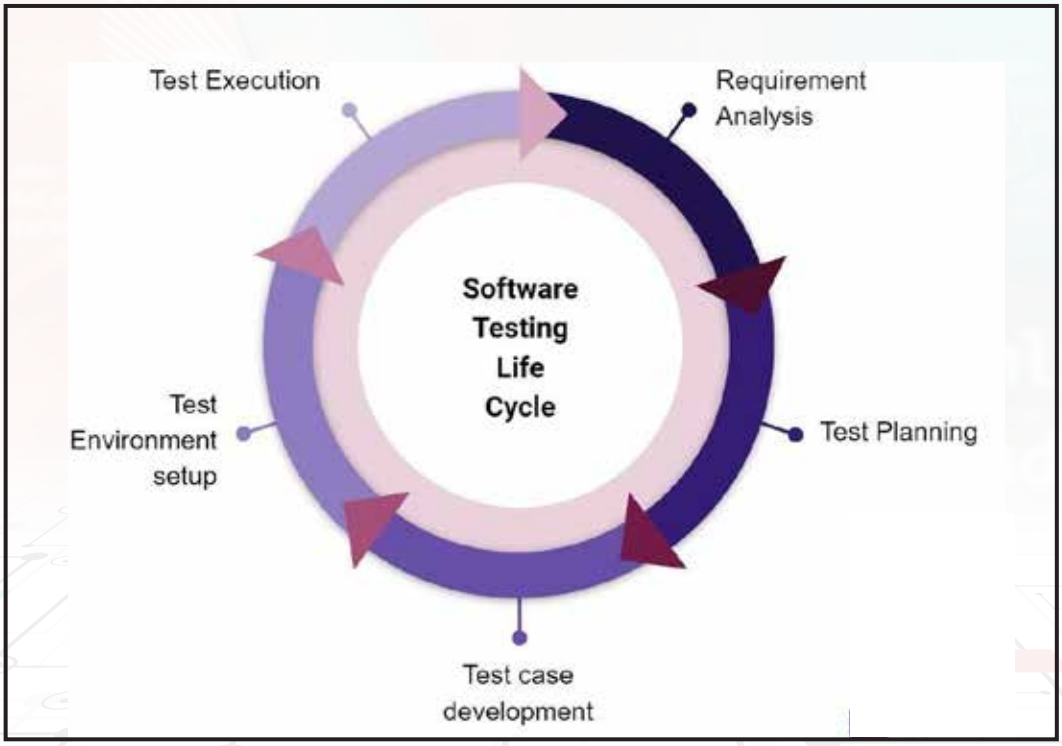


Table 11.0.2 Software Testing Life Cycle

1. Requirement Phase:

During this phase, QA team studies the requirements from a testing point of view to identify the testable requirements.

- Identify types of tests to be performed.
- Gather details about testing priorities and focus.
- Prepare Requirement Traceability Matrix (RTM).
- Identify test environment details where testing is supposed to be carried out.

2. Test Planning Phase:

Identify the activities and resources which would help to meet the testing objectives.

- Preparation of test plan/strategy document for various types of testing.
- Test tool selection.
- Test effort estimation.
- Resource planning and determining roles and responsibilities.
- Training requirement.

3. Test Case Development Phase:

This phase involves the creation, verification and rework of test cases & test scripts.

- Create test cases, automation scripts
- Review and baseline test cases and scripts
- Create test data (If Test Environment is available)

4. Test Environment Setup Phase:

Based on the Customer's environment Test environment setup for the software and hardware conditions of the product.

- Prepare hardware and software requirement list for the Test Environment.
- Setup test Environment and test data.
- Perform Smoke Test (verified to ensure the stability of the application for further testing) on the build.

5. Test Execution Phase:

Testing carried out based on the test plans and the test cases. Bugs will be reported back to the development team for correction and retesting will be performed.

- Execute tests as per plan.
- Document all the test results, and log defects for failed cases.
- Map defects to test cases in Requirement Traceability Matrix (RTM).
- Retest once the development team fixed the defects.

6. Test Cycle Closure Phase:

QA team analyze testing artifacts and performance of the testing to identify new strategies for the future testing. This is a learning process.

- Evaluate cycle completion criteria based on Time, Test coverage, Cost, Software, Critical Business Objectives, Quality.
- Document the learning outcomes from the software.

How to write a Test Case?

Test cases help guide the QA team through a sequence of steps to validate whether a software application is free of errors and working as required by the end user. The test case includes specific variables or conditions, so the QA team can compare expected and actual results to determine whether a software is functioning as the end user's requirements.

Test Cases for Login Page

Figure 11.0.3 Test Cases for Login Page

The format of Standard Test Cases

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail

Table 11.0.3 Format of Standard Test Cases

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
T01	Verify that cursor is focused on "Username" text box on the page load	After Loading the Login Page	-	Cursor if focused on username field	As Expected	Pass
T02	Verify that the Login screen contains elements such as Username, Password, Login button, Forgot Username/ Password link	After Loading the Login Page	-	Verify the components in Login screen	As Expected	Pass
T03	Verify that Tab functionality is working	Press the tab button	-	Cursor moves to	As Expected	Pass

	properly or not			Password field		
T04	Verify that the password is in masked form when entered	Enter some dummy values	1234	Password field is masked by *	As Expected	Pass
T05	Verify if the password can be copy-pasted or not.	Copy paste a value to the password field	1234	Cannot paste the value to the password field	Not as Expected	Fail
T06	Verify that User is able to Login with Valid Credentials	Enter Email and Password . Then click Login button	Email= keshan@gmail.com Password= lec789	User should Login into the system	As Expected	Pass
T07	Verify that User is not able to Login with invalid Username and invalid Password	Enter Email and Password . Then click Login button	Email= kehan@gmail.co Password= lec1234	Display Error Message Your Email or Password is Wrong	As Expected	Pass
T08	Verify that User is not able to Login with Valid Username	Enter Email and Password . Then click	Email = keshan@gmail.com Password = 45789	Display Error Message Your Email or Password is Wrong	As Expected	Pass

	and invalid Password	Login button				
T09	Verify that User is not able to Login with invalid Username and Valid Password	Enter Email and Password . Then click Login button	Email = keshan@gmail.com Password = lec789	Display Error Message Your Email or Password is Wrong	As Expected	Pass
T10	Verify that User is not able to Login with blank Username or Password	Enter Email and Password . Then click Login button	Email = Password = 45789	Display Error Message Please enter your	As Expected	Pass
T11	Verify that User is not able to Login with inactive credentials	Enter Email and Password . Then click Login button	Email = keshan@gmail.com Password = lec789	Display Error Message Your account is Inactive	As Expected	Pass
T12	Verify that there is a only 3 unsuccessful login attempts	Enter Email and Password . Then click Login button	Email = kes@gmail.com Password = 45789	Display Error Message Your Login attempts are over. Your	As Expected	Pass

				account Lock		
T13	Verify that User should be able to login with the new password after changing the password	Enter Email and Password . Then click Login button	Email = keshan@gmail.com Password = lec123	User should Login into the system	As Expected	Pass
T14	Verify that User should not be able to login with the old password after changing the password	Enter Email and Password . Then click Login button	Email = kes@gmail.com Password = lec789	Display Error Message Your Email or Password is Wrong	As Expected	Pass
T15	Verify that User is redirected to appropriate page after successful login	Enter Email and Password . Then click Login button	Email = keshan@gmail.com Password = lec123	User should able to see the Home Menu	As Expected	Pass

Table 11.4 Test Cases for Login Page

Software Testing Techniques (Methods)

Black box testing use to test the software without knowing the internal structure of code or program

White box testing use to test the software with the knowledge of internal structure of code or program

Black Box Testing

- Also known as Functional Testing. Programming knowledge or Implementation knowledge is not required.
- The3 main aim of this testing to check on what functionality is performing by the system under test.
- The black box testing is used to find,
    - Interface errors
    - Incorrect Functions that lead to undesired output
    - Missing Functions
    - Incorrect Outputs

An Example for Black Box Testing

- Testing User Name and Password in a simple Login screen.
- This test will check the input and output.

- A user can log to the system when inputs the correct username and correct password.
- A user receives an error message when enters incorrect username and incorrect password.

Figure 11.0.4 Black Box Testing - Example

White Box Testing

- Also known as Structural Testing or Glass Box Testing. Knowledge of internal working structure (Coding of software) is required.
- In White Box testing is primarily concentrate on the testing of program code of the system under test like code structure, branches, conditions, loops etc.
- The white box testing is used to find
  - Internal Security Holes
  - Testing of each statement
  - Functionality of conditional loops

Black Box Testing Technique - Equivalence Partitioning

This allows QA team to separate test conditions into partitions (classes). Test cases should be designed according to the different input domains created.

Ex: Suppose you want to test a input box which is accepting numbers from 1 to 100. There is no use in writing 100 test cases for all 100 valid input numbers plus other test cases for invalid data.

Using the Equivalence Partitioning method above test cases can be divided into three sets of input data called classes.

Testing Solution:

1. One input data class with all valid inputs. You can take a single value from range 1 to 100 as a valid test case.

2. Input data class with all values below the lower limit. Any value below 1, as an invalid input data test case.

3. Input data with any value greater than 100 to represent the third invalid input class.

Black Box Testing Technique - Boundary Value Analysis

Boundary testing is the process of testing between extreme ends or boundaries between partitions (classes) of the input values. More software **errors occur at the boundaries** of the input partitions. 'Boundary Value Analysis' Testing technique is used to identify errors at boundaries.

Ex: Suppose you want to test an input box which is accepting numbers from 1 to 100.

In this case you can test boundary values using boundary value analysis.

Testing Solution:

1. Test exactly the input boundaries of input domain. Test value 1 and 100.

2. Test data with values just below the extreme edges of input domains. Test 0 and 99.
3. Test data with values just above the extreme edges of the input domain. Test 2 and 101.

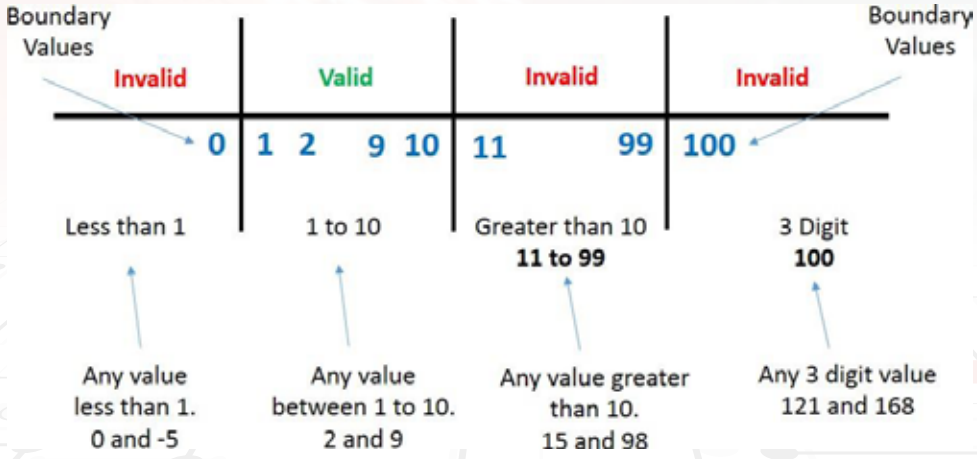
Example for Equivalence Partitioning & Boundary Value Analysis

Consider about a text box which accepts number of pizza. Maximum 10 pizza can be order and 1 to 10 is considered as valid inputs.

Here are the test conditions

- Any Number greater than 10 is invalid.
- Any Number less than 1 is invalid.
- Numbers 1 to 10 are considered valid
- Any 3 Digit Number say 100 is invalid.

We cannot test all the possible numbers. Then the number of test cases will be more than 100. Ex: You have to test 1 to 10. Then values greater than 10, less than 1, more than 2-digit number. Therefore, we can use Equivalence Partitioning technique.



Software Testing Levels (Types)

Functional Testing Types

- Unit Testing
- Integration Testing
- System Testing
- Sanity Testing
- Smoke Testing
- Interface Testing
- Regression Testing
- Beta/Acceptance Testing

Non Functional Testing Types

- Performance Testing
- Load Testing
- Stress Testing
- Volume Testing
- Security Testing
- Compatibility Testing
- Install Testing
- Recovery Testing
- Reliability Testing
- Usability Testing
- Compliance Testing
- Localization Testing

Functional Testing Types

Unit Testing

Test individual units/components of the software. The purpose is to validate that each unit or component of the software performs as designed. It is typically done by the Software Engineer and not by QA Engineer.

Integration Testing

Individual units are combined and tested as a group. The aim of this level is to expose errors in the interaction between integrated units.

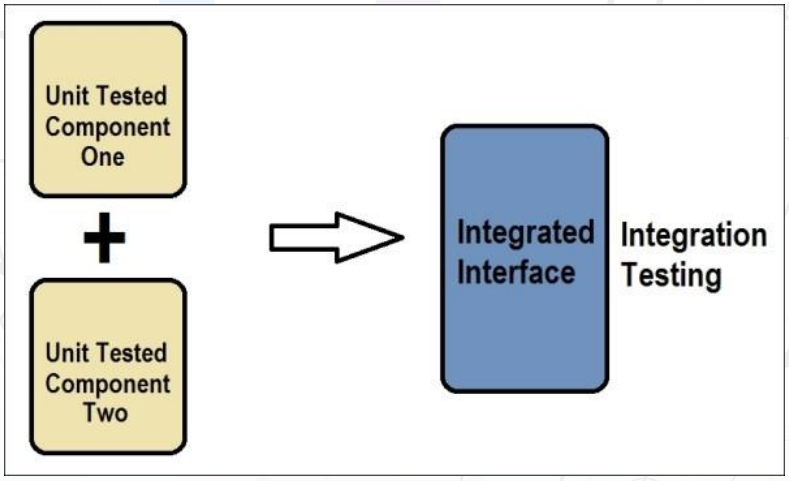


Figure 11.0.5 Integration Testing

System Testing

The entire software is tested as per the requirements specification. It is a Black-box type testing that is based on overall requirement specifications and covers all the combined parts of a system.

Smoke Testing

Whenever a new build is provided by the development team then the QA team validates the build to ensures that no major issue exists. This testing is ensuring that the build is stable and a detailed level of testing can be carry out further.

If QA team find that the major critical functionality is broken down at the initial stage they reject the build

Acceptance (Alpha) Testing

Alpha testing is a type of acceptance testing which is perform to identify all possible bugs before releasing the software to real environment. This test is performing in the development environment by the developers and QA team (in the development site).

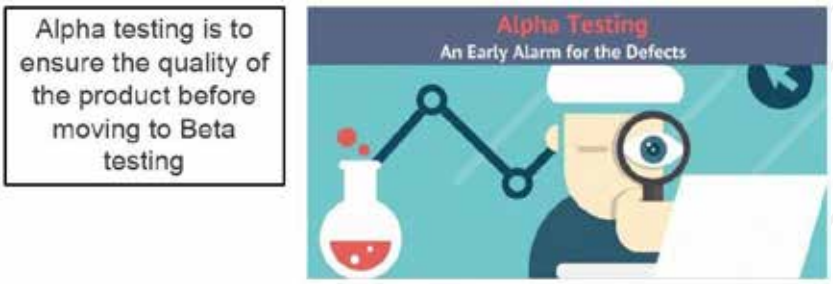


Figure 11.0.6 Acceptance (Alpha) Testing

Acceptance (Beta) Testing

An Acceptance Test is performed by the customer and users to verifies whether the system build as per the business requirements. This test is performing in the real environment by the real users (in the Client site).

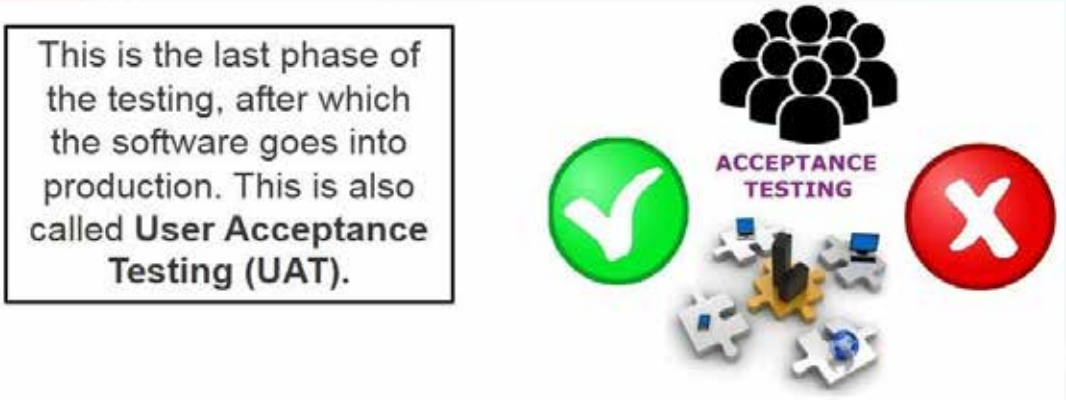


Figure 11.0.7 Acceptance (Beta) Testing

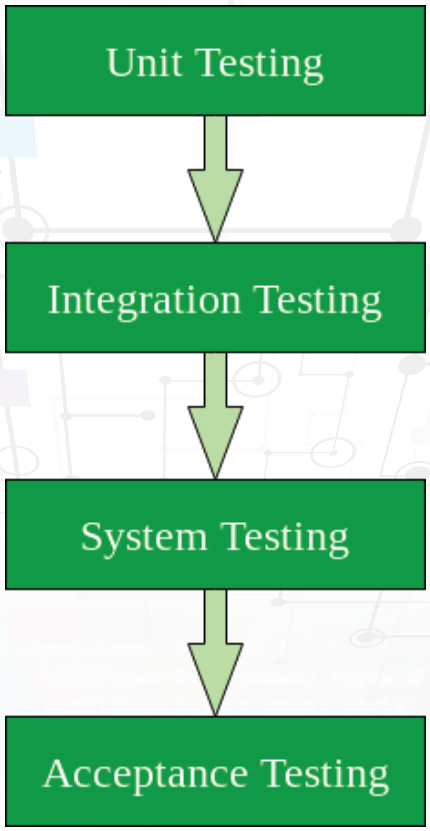


Figure 11.0.8 Software Testing Levels

Non Functional Testing Types

Performance Testing

- Performance Testing is done to check whether the system meets the performance requirements. Different performance and load tools are used to do this testing.

Load Testing

- To check how much load or maximum workload the software can handle without any performance deviations.

Stress Testing

- Software is stressed beyond its specifications in order to check how and when it fails.
- This is performed under heavy load like putting large number beyond storage capacity, complex database queries, continuous input to the system or database load.

Security Testing

- To check how the software is secure from internal and external threats.
- This testing includes how much software is secure from the malicious program, viruses and how secure and strong the authorization and authentication processes are.

Compatibility Testing

- This testing validates how software behaves and runs in a different environment, web servers, hardware, and network environment.

Exhausting Testing

Exhaustive testing is a test approach in which all possible data combinations are used testing.

In this method all the types of inputs are given to the software to check the outputs.

Exhaustive testing is the process of testing for absolutely everything just to make sure that the software won't crash in any situation.

Ex: Consider the password field which accepts 3 characters only.

- If you take the alphabet only. You need to check 26 * 26 * 26 inputs
- If you take numbers and special characters also. It is 256 * 256 * 256 inputs

Exhausting Testing is not possible for complex large project. You cannot test all the possible inputs and combinations

Top Testing Tools

- **Bugzilla:**

  It is a Bug-tracking system which allow QA team to keep track of outstanding bugs, problems, issues, enhancement and other change requests in their products effectively.

- **Selenium:**

  Selenium is a testing framework to perform web application testing across various browsers and platforms like Windows, Mac, and Linux.

- **TestRail:**

  Efficiently manage manual and automated test cases, plans, and runs. Get real-time insights into testing progress with informative dashboards, metrics, and activity reports.

- **Loadrunner:**

  It is a load testing tool for Windows and Linux, which allows testing the web application efficiently. It helpful testing tool to determining the performance and result of the web application under heavy load.