Testing:

Process to find errors in a program/product and to detect errors/bugs/defect

Verification – the process of checking that the software/ application meets the Specifications. Verification uses methods like reviews, walkthroughs, inspections and desk- checking etc.

Validation – It checks whether software meets the requirements and expectations of customer. It uses methods like [Black Box Testing](https://www.guru99.com/black-box-testing.html),[White Box Testing](https://www.guru99.com/white-box-testing.html) and non-functional testing

Verification – Are we building the product right?

• Validation – Are we building the right product?

Black Box Testing: Testing a system without having specific knowledge to the internal workings of the system. No access to the source code.

White Box Testing: testing a system with full knowledge and access to all the source ode and architecture documents. So, can catch bugs more quickly rather than “trail and error ” method of black box testing. Additionally, we can be sure to get more complete testing coverage by knowing exactly what you have to test

Grey Box Testing: Testing a system while having atleast some knowledge of the internal of a system. This knowledge is usually contrained to detailed design documents and architecture diagrams.

Types of testing:

* Functional
* Non - Functional
* Maintenance

**Functional:**

**Unit testing** - done by developer

First level of testing, most micro-level of testing

**Integration Testing** – can be done by either qa or a developer

Can be black box, white box or a grey box

Webservice, api testing, etc

Top down, bottom up and big bang Testing approaches

The top down approach starts by testing the top-most modules and gradually moving down to the lowest set of modules one-by-one.

The bottom up approach starts with testing the lowest units of the application and gradually moving up one-by-one.

Big bang: prioritize and work in both directions

Smoke/Sanity

[Smoke Testing](https://www.guru99.com/smoke-testing.html) is a kind of Software Testing performed after software build to ascertain that the critical functionalities of the program is working fine. It is executed "before" any detailed functional or regression tests are executed on the software build. The purpose is to reject a badly broken application, so that the QA team does not waste time installing and testing the software application.

## Smoke Testing Vs Sanity Testing - Key Differences

|  |  |
| --- | --- |
| **Smoke Testing** | **Sanity Testing** |
| Smoke Testing is performed to ascertain that the critical functionalities of the program is working fine | Sanity Testing is done to check the new functionality / bugs have been fixed |
| The objective of this testing is to verify the "stability" of the system in order to proceed with more rigorous testing | The objective of the testing is to verify the "rationality" of the system in order to proceed with more rigorous testing |
| This testing is performed by the developers or testers | Sanity testing is usually performed by testers |
| Smoke testing is usually documented or scripted | Sanity testing is usually not documented and is unscripted |
| Smoke testing is a subset of [Regression Testing](https://www.guru99.com/regression-testing.html) | Sanity testing is a subset of Acceptance testing |
| Smoke testing exercises the entire system from end to end | Sanity testing exercises only the particular component of the entire system |
| Smoke testing is like General Health Check Up | Sanity Testing is like specialized health check up |

**User Acceptance, etc**

The final level of testing, acceptance testing, or [UAT (user acceptance testing)](https://blog.testlodge.com/what-is-user-acceptance-testing/), determines whether or not the software is ready to be released.

User verifies the business needs are met *before* the software is released into production.

Alpha testing: done by qa, stake holder, developers environment

Beta testing: customers environment, done by end users

**NonFunctional**

Performance

Load

Volume

Scalability

Usability

**Maintenance:**

Regression

SDLC -> software development life cycle

* set of phases that provide a model for development and life cycle management of application or a software

1. Planning
2. Analysis
3. Design
4. Implement
5. Testing
6. Maintenance

Goal -> to develop and maintain the product/ application

STLC -> Software Test Life Cycle

1. Requirement Analysis
2. Test Planning
3. Test case development
4. Environment setup
5. Test execution
6. Test cycle closure

## Requirement Analysis

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

The QA team may interact with various stakeholders (Client, Business Analyst, Technical Leads, System Architects etc) to understand the requirements in detail.

Requirements could be either Functional (defining what the software must do) or Non Functional (defining system performance /security availability )

## Test Planning

This phase is also called **Test Strategy** phase. Typically , in this stage, a Senior QA manager will determine effort and cost estimates for the project and would prepare and finalize the Test Plan.

## Test Case Development

This phase involves creation, verification and rework of test cases & test scripts. [Test data](https://www.guru99.com/software-testing-test-data.html" \t "_blank) , is identified/created and is reviewed and then reworked as well. **Activities**

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

**Test Case template**

* 1. Test case iD
* 2. Description
* 3. Step details
* 4. Expected results
* 5. actual results
* 6. pass/fail/
* 7. defect id

## Test Environment Setup

Test environment decides the software and hardware conditions under which a work product is tested. Test environment set-up is one of the critical aspects of testing process and ***can be done in parallel with Test Case Development Stage***. ***Test team may not be involved in this activity*** if the customer/development team provides the test environment in which case the test team is required to do a readiness check (smoke testing) of the given environment.

## Test Execution

 During this phase the testers will carry out the testing based on the test plans and the test cases prepared. Bugs will be reported back to the development team for correction and retesting will be performed.

Requirements -> analyse end user requirements and remove any inconsistency. Analyse adv and dis adv of requirements and make dev of product/ application cost effective

Product owner: Interface btwn client and the dev team

Reviews/complaints: Product feedback

Testing: test the application against given requirements to assure quality

Integration, acceptance,

Maintenance- product support/operations to provide support wrt app for end user

Design -> SRS Documents (Software requirement specifications)

BRD Documents (Business requirement documents)

All the details are specified here. Scope, objective, etc

Agile Principles:

Less documentation and more work

Business people and developers work together

Working software is the primary measure

Time critical applications

Agile Methods:

Scrum -> story points

Xp(extreme programing) – more and more programing.

Test driven development, tools and there are many other processes that implement agile methodology as a framework

Terms:

Sprint – duration in which we set requirements, develop, test and deliver.

Divide the project in terms of time line. (1-4 weeks)

Product Owner – represents end user or he is the voice of the business owner

Scrum master – project manager. Coordinates btwn developers, testers

Scrum team – 7+-2

4 developers, 2 qa and 1 lead

User story – Specific requirements divided to team members or an individual requirement. Generally created by product owner

Daily standup meeting – 15-20 mins every day

Whats done so far, any issues to be discussed, what you are going to do for that day.

Sprint planning meeting – before sprint starts, to finalize on sprint backlog, estimations for story points, etc

Sprint review meeting – at the end of sprint, to demo the app developed so far to entire team ie product review

Sprint Retrospective meeting – process review so that in the next sprint we don’t want to repeat same errors

Velocity – how many user-stories the team can complete in a given time

Story grooming – what are the requirements, functional way of requirements

Story planning – based on team velocity, based on team past, how much time the team need to deliver the product

Release – move the application code into prod server/go live.

**Defect Life Cycle:**

The number of states that a defect goes through varies from project to project. Below lifecycle diagram, covers all possible states

* **New:** When a new defect is logged and posted for the first time. It is assigned a status NEW.
* **Assigned:** Once the bug is posted by the tester, the lead of the tester approves the bug and assigns the bug to developer team
* **Open**: The developer starts analyzing and works on the defect fix
* **Fixed**: When developer makes necessary code change and verifies the change, he or she can make bug status as "Fixed."
* **Pending retest**: Once the defect is fixed the developer gives particular code for retesting the code to the tester. Since the testing remains pending from the testers end, the status assigned is "pending request."
* **Retest**: Tester does the retesting of the code at this stage to check whether the defect is fixed by the developer or not and change the status to "Re-test."
* **Verified:** The tester re-tests the bug after it got fixed by the developer. If there is no bug detected in the software, then the bug is fixed and the status assigned is "verified."
* **Reopen**: If the bug persists even after the developer has fixed the bug, the tester changes the status to "reopened". Once again the bug goes through the life cycle.
* **Closed**: If the bug is no longer exists then tester assigns the status "Closed."
* **Duplicate**: If the defect is repeated twice or the defect corresponds the same concept of the bug, the status is changed to "duplicate."
* **Rejected**: If the developer feels the defect is not a genuine defect then it changes the defect to "rejected."
* **Deferred**: If the present bug is not of a prime priority and if it is expected to get fixed in the next release, then status "Deferred" is assigned to such bugs
* **Not a bug**:If it does not affect the functionality of the application then the status assigned to a bug is "Not a bug".

While opening a bug, points to remember:

Description

Actual

Expected

Attachements

Logfile

Priority

Iteration/sprint

Release

Replication specs

Priority -> how quickly the defect should be fixed

Severity -> how it is effecting/ impacting business, financial loss, etc

Eg: High Priority, high severity

Cart Fucntionality not working in ecommerce

Low priority, high severity

Some where in navigation links, one of the links not working/logging out of the appl

Low priority, low severity

Some typos in rarely visited screens

High priority, low severity

Company logo/ name incorrect