'**software**' refers to the set of electronic program instructions or data a computer processor reads in order to perform a task or operation.

There are two main types of software: **systems software** and **application software**.

**Systems Software**

**Systems software** includes the programs that are dedicated to managing the computer itself, such as the **operating system**, file management utilities, and disk operating system (or DOS). The operating system manages the computer hardware resources in addition to applications and data. Without systems software installed in our computers we would have to type the instructions for everything we wanted the computer to do

## Applications Software

**Application software**, or simply **applications**, are often called productivity programs or end-user programs because they enable the user to complete tasks, such as creating documents, spreadsheets, databases and publications, doing online research, sending email, designing graphics, running businesses, and even playing games! Application software is specific to the task it is designed for and can be as simple as a calculator application or as complex as a word processing application.

**The different types of application software include the following:**

| **Application Software Type** | **Examples** |
| --- | --- |
| Word processing software | MS Word, WordPad and Notepad |
| Database software | Oracle, MS Access etc |
| Spreadsheet software | Apple Numbers, Microsoft Excel |
| Multimedia software | Real Player, Media Player |
| Presentation Software | Microsoft Power Point, Keynotes |
| Enterprise Software | Customer relationship management system |
| Information Worker Software | Documentation tools, resource management tools |
| Educational Software | Dictionaries: Encarta, BritannicaMathematical: MATLABOthers: Google Earth, NASA World Wind |
| Simulation Software | Flight and scientific simulators |
| Content Access Software | Accessing content through media players, web browsers |
| Application Suites | OpenOffice, Microsoft Office |
| Software for Engineering and Product Development | **IDE** or Integrated Development Environments |

### What is Software Testing?

## Why is Software Testing Important?

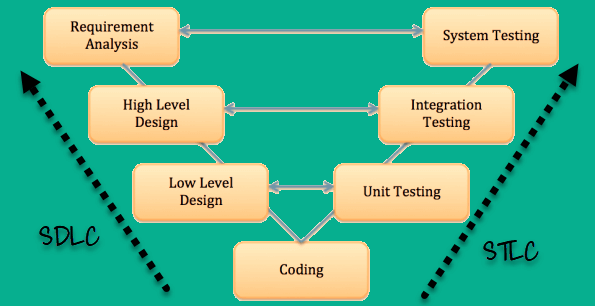
## Who does Testing?

* Software Tester
* Software Developer
* End User

## When to Start Testing?

Testing is done in different forms at every phase of SDLC −

* During the requirement gathering phase, the analysis and verification of requirements are also considered as testing.
* Reviewing the design in the design phase with the intent to improve the design is also considered as testing.
* Testing performed by a developer on completion of the code is also categorized as testing.



Software Test Life Cycle has the following stages.

1. Requirement Analysis
2. Test Planning
3. Test Development
4. Test Environment Setup
5. Test Execution & Closure

|  |  |  |
| --- | --- | --- |
| **Parameter** | **SDLC** | **STLC** |
| Origin | Development Life Cycle | Testing Life Cycle |
| Objective | The main object of SDLC life cycle is to complete successful development of the software including testing and other phases. | The only objective of the STLC phase is testing. |
| Requirement Gathering | In SDLC the business analyst gathers the requirements and create Development Plan | In STLC, the QA team analyze requirement documents like functional and non-functional documents and create System Test Plan |
| High & Low-Level Design | In SDLC, the development team creates the high and low-level design plans | In STLC, the test analyst creates the Integration Test Plan |
| Coding | The real code is developed, and actual work takes place as per the designd documents. | The testing team prepares the test environment and executes them |
| Maintenance | SDLC phase also includes post-deployment supports and updates. | Testers, execute regression suits, usually automation scripts to check maintenance code deployed. |

|  |  |  |
| --- | --- | --- |
| **Sr.No.** | **Verification** | **Validation** |
| 1 | Verification addresses the concern: "Are you building it right?" | Validation addresses the concern: "Are you building the right thing?" |
| 2 | Ensures that the software system meets all the functionality. | Ensures that the functionalities meet the intended behavior. |
| 3 | Verification takes place first and includes the checking for documentation, code, etc. | Validation occurs after verification and mainly involves the checking of the overall product. |
| 4 | Done by developers. | Done by testers. |
| 5 | It has static activities, as it includes collecting reviews, walkthroughs, and inspections to verify a software. | It has dynamic activities, as it includes executing the software against the requirements. |
| 6 | It is an objective process and no subjective decision should be needed to verify a software. | It is a subjective process and involves subjective decisions on how well a software works. |

## When to Stop Testing?

It is difficult to determine when to stop testing, as testing is a never-ending process and no one can claim that a software is 100% tested. The following aspects are to be considered for stopping the testing process −

* Testing Deadlines
* Completion of test case execution
* Completion of functional and code coverage to a certain point
* Bug rate falls below a certain level and no high-priority bugs are identified
* Management decision

1. **Levels Of Testing**

**Functional Testing**

1. **Unit testing**
2. **Integration testing**
3. **Smoke testing**
4. **Sanity testing**
5. **System testing**
6. **Regression testing**
7. **Acceptance testing(alpha, beta)**
8. **E2E testing**
9. **Adhoc testing**
10. **Retesting**

**Non-Functional Testing**

1. **Performance Testing(Load, Stress, Volume, Spike)**
2. **Usability Testing**
3. **Accessibility Testing**
4. **Security Testing**
5. **Methods of Testing**
6. **Black Box Testing**
7. **White Box Testing**
8. **Grey Box Testing**

**Test Management**

* 1. **Bug life Cycle**