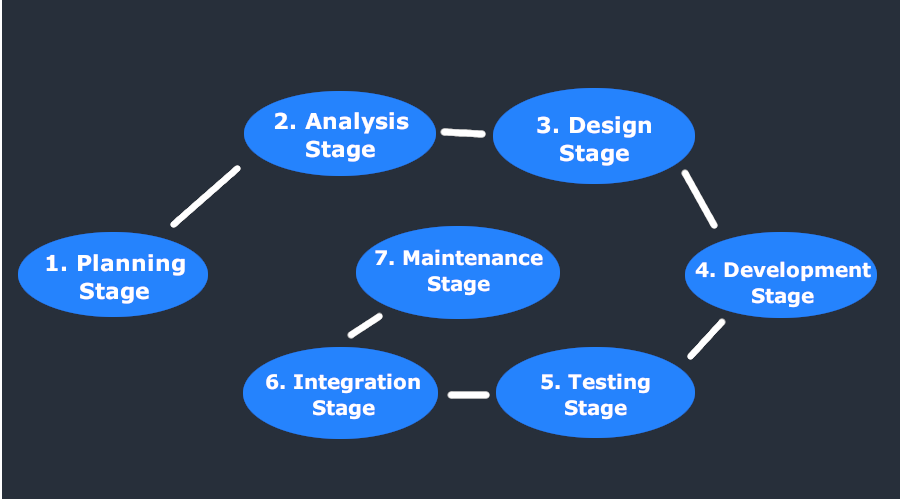
**SDLC**

Includes 7 cycles/phases

1. Planning
2. Requirements Analysis
3. Design
4. Implementation/Coding
5. **Testing**
6. Deployment
7. Maintenance



**STLC**

Software Testing Lifecycle

Includes the following cycles/phases:

1. Requirement analysis
2. Test planning phase
3. Test case design and development
4. Test environment setup
5. Test execution
6. Test reportingd
7. Test closure
   * In the closure documentation, you write what the complete testing included, what are the artifacts, etc.

**7 Testing Principles**

1. Testing shows presence of defects
   * Even multiple testing can never ensure that software is 100% bug-free
   * Software testing talks about the presence of defects and not about the absence of defects
2. Exhaustive testing is not possible
3. **Early Testing** should be preserved
4. **Defect Clustering** 
   * Small number of modules contain most of defects detected
   * Pareto principle to software testing state that 80% of software defects comes from 20% of modules
5. **Pesticide Paradox**
   * If the same tests are repeated over and over again, eventually the same test cases will no longer find new bugs.
   * Therefore, the test cases need to be regularly reviewed and revised, adding new and different test cases to help find more defects
6. Testing is context dependent
   * The testing of e-commerce site is different from the testing of an Android application
7. Absence of error is a fallacy

**Software Testing Levels**

Software testing levels **IS NOT EQUAL TO** software testing types

**4 levels of testing:**

1. Unit testing
   * Also called Component testing
   * Standalone unit/component being tested
   * Mainly responsibly of the developer
2. Integration testing
   * Multiple units integrated together into one module, e.g., login form/page with all units (password, username, etc.)
   * Responsibility of the testing team
3. System testing
   * Testing the overall application/software when developed (e.g., an e-commerce website)
   * System testing is divided into:
   1. System test
      * System standalone as a whole
      * Doesn’t integrate with another system
   2. System integration test
      * The software is not standalone and integrates with another external system
4. User acceptance testing (UAT)
   * Done by the business or the customer who is using the software
   * Tests whether the software provides the functionality the customer is looking for
   * The consumer tests whether the software that has been built is fit for use

**Software Testing Types**

There are many types of testing according to Google, however there are 2 main categories of testing types:

* **Functional Testing** -> Testing of the functionality of a particular software
  + Unit testing
  + Integration testing
  + System testing
  + User acceptance testing
  + Regression testing
  + Sanity testing
  + Smoke testing
  + Usability testing
* **Non-functional testing** -> Testing refresh rate is an example of non-functional testing
  + Performance testing
  + Load testing
  + Stress testing
  + Volume testing
  + Scalability testing
  + Recovery testing
  + Compatibility testing
  + Security testing

**Black Box Testing**

You don’t know the internal structure of the system – it is a black-box.

You provide the input and just verify the output.

Basically the functionality is being tested.

Diagram

Description automatically generated

You are aware of the internal structure of the system and what happens step by step behind the scenes.

White-box testing is a method of software testing that tests internal structures or workings of an application, as opposed to its functionality.

**Equivalence Partitioning**

Equivalence partitioning is a software black-box testing technique that divides the input data of a software unit into partitions of equivalent data from which test cases can be derived.

Example:

A picture containing text, whiteboard

Description automatically generated

Sales divided into categories of discounts. Imagine you test the discount partitions (5%, 10%, 20%, etc.). All these discount values are considered equivalent partitions. The discount module accepts item price in dollars as input. Each of these partitions provide test coverage. Equivalent partitioning is a technique to come up with a minimal number of test cases which will give you a very broad test coverage.

**Boundary Value Analysis**

Boundary-value analysis is a software black-box testing technique in which tests are designed to include representatives of boundary values in a range. We then come up with test cases within those boundaries. You have to be careful with decimal values (99 or 99.99) as a boundary value. Together with equivalent partitioning, you have a pre-defined test cases in the face of boundary values.

