FIRST DAY TOPICS:

**Return on investment** (**ROI**)

 ROI tries to directly measure the amount of return on a particular investment, relative to the investment’s cost. To calculate ROI, the benefit (or return) of an investment is divided by the cost of the investment. The result is expressed as a percentage or a ratio.

**return on investment** = Net income / Investment

where:

Net income = gross profit − expenses.

investment = stock + market outstanding [claim

**Capability Maturity Model** (**CMM**)

it is used as benchmark to measure the maturity of an organization in software development process.

There are levels in cmm to implement

1.Initial

2. Repeatable

3.Defined

4.Quantitatively managed

5.Optimizing

**INITIAL**:

* In the initial level there is no standard process for the software development system.
* it does not containing the project tracing system that enables the developers to predict the product cost and estimate the project duration.
* it is not having the stable environment.

**Repeatable;**

* In this stage the basic software and management process control will be available in planning and tracking.
* Estimation of time duration and it works is planned and traced.

**DEFINED:**

* Makes sure that product meets the requirements and intended use

**Quantitatively managed:**

* **The Quantitative** objectives are useful for quality and process performance, that are established and used as a criteria in **managing** processes.
* **Quantitative** objectives are based on the needs of the customer, end users, organization, and process implementers

**OPTIMIZATION**

* it focus on the innovations, improving the processes through the both incremental and decremental process.

**Quality Assurance and Quality Control**

* The difference between quality assurance and quality control is that Quality Control is product oriented, while Quality Assurance is process oriented.
* QC is a testing checks the quality of a product and it classifies under the domain of QC. When you are testing a product for quality, you are not assuring its quality, you are controlling it.
* QA makes sure that what you are doing are right things in the right manner
* While QC ensures the results of what you have done are as per your expectation
* Both QC and QA are interdependent to each other

## **SDLC models with pros and cons: (waterfall, V model, spiral, Agile Scrum)**

# **Waterfall:**

Waterfall Model is among the traditional SDLC methodologies. In fact, it is the oldest, most rigid and even easiest to understand. The idea here is quite simple: develop one phase of the project completely and move on to the next. Develop the new phase and advance to the next one.

**Pros:**

* Waterfall model is easy to understand due to its simple linear structure
* It helps to define the goals and deliverables at the early stage of the project
* Project management is simple and effective as there are no unnecessary changes

**Cons:**

* Impossible to make changes at the later stage
* This SDLC model doesn’t work for maintenance type project
* No working model of the software until the end of the life cycle

# **V-Model:**

In this model V stands for verification and validation and shape of this model is also V because some testing stages depends on documents, here testing comes in every stage of development.

V-model is suitable for software development of sectors like

* Banking, finance services
* E-commerce
* Healthcare
* Logistics etc.

The cons in the waterfall model are over covered here.

# **Spiral Model:**

Spiral Model is considered to be one of the best Software Development Life Cycle methodologies for a large and complex project. It is an amalgamation of the two methodologies Waterfall Model and Prototyping Model.

**Pros:**

* The spiral model is known for its flexibility as changes can be made on the later stage of the project
* It is one the Software Development Life Cycle Methodologies that helps in mitigating the risk by analysing and solving it beforehand
* Clients are connected with the development team and project as they share the requirements and feedbacks of each phase
* It is suitable for long-term and complex projects

**Cons:**

* Requires high risk-analysing expertise
* The project takes a significantly long time to develop, increasing the overall expense of the project
* It is complex to understand and implement
* Since the number of iterations are unknown, the time required to complete the project remains a mystery

# **Agile Scrum:**

Agile method is an incremental and iterative approach to software development, it follows a work breakdown structure. The whole product is broken down into smaller logical segments and worked on sections in an incremental fashion.

**PRODUCT**

**EPIC**

**USERSTORIES**

**TASK**

**Scrum team:**

The scrum team has three important roles,

* Product owner
* Scrum Team
* Scrum master

In Agile scrum a big project is divided into small tasks called as ‘**sprints**’, each sprint has a deadline of 2-3 weeks,

The **product owner** gives the **user stories** and the scrum team included with **Developers** and **Testers** work on their respective sprints, and all these are done under the support of the **Scrum master**, he/her can monitor the work done by using.

\*These days Agile Scrum is the most used software development methodology in market

**Software Configuration Management:**

Software Configuration Management is defined as a process to systematically manage, organize, and control the changes in the documents, codes, and other entities during the Software Development Life Cycle. It is abbreviated as the SCM process in software engineering. The primary goal is to increase productivity with minimal mistakes.

**The important reasons for Implementing Software Configuration Management System are,**

* There are multiple people working on software which is continually updating
* It may be a case where multiple version, branches, authors are involved in a software project, and the team is geographically distributed and works concurrently
* Changes in user requirement, policy, budget, schedule need to be accommodated.
* Software should able to run on various machines and Operating Systems
* Helps to develop coordination among stakeholders
* SCM process is also beneficial to control the costs involved in making changes to a system

## **Software architecture**

Software architecture refers to the fundamental structures of a software system and the discipline of creating such structures and systems. Each structure comprises software elements, relations among them, and properties of both elements and relations.

A picture containing electronics

Description automatically generated

## **What Is Software Testing Life Cycle (STLC)?**

Software Testing Life Cycle refers to a testing process which has specific steps to be executed in a definite sequence to ensure that the quality goals have been met.

In the STLC process, each activity is carried out in a planned and systematic way. Each phase has different goals and deliverables. Different organizations have different phases in STLC; however, the basis remains the same.

**Below are the phases of STLC:**

1. Requirements phase
2. Planning Phase
3. Design Phase
4. Test environmental setup
5. Implementation Phase
6. Closure Phase

[**Requirement Phase**](https://www.softwaretestinghelp.com/rview-srs-document-and-create-test-scenarios-software-testing-training-course-day-2/)**:**

Requirement Analysis is the first step in Software Testing Life Cycle.

In this phase quality assurance team understands the requirements like what is to be tested.

If anything is missing, then quality assurance team meets with the stakeholders to better understand the detail knowledge of requirement

[**Planning Phase**](https://www.softwaretestinghelp.com/how-to-write-test-plan-document-software-testing-training-day3/)

Test Planning is most efficient phase of software testing life cycle where all testing plans are defined.

In this phase manager of the testing team calculates estimated effort and cost for the testing work.

This phase gets started once the requirement gathering phase is completed.

**Design Phase:**

**design phase** is the process of creating and writing **test**s for **testing** a software.

**Test** analysis and identifying **test** conditions gives us a generic idea for **testing** which covers quite a large range of possibilities.

**Test Environment Setup:**  
Test environment setup is the vital part of the STLC. Basically, test environment decides the conditions on which software is tested.

This is independent activity and can be started along with test case development.

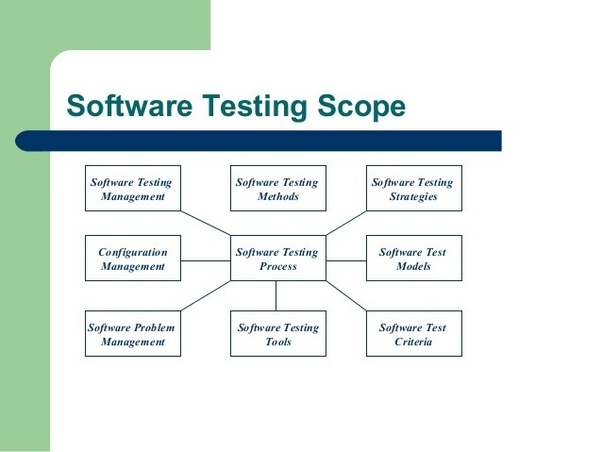
**Implementation Phase:**

After the test case development and test environment setup test execution phase gets started. In this phase testing team start executing test cases based on prepared test cases in the earlier step.

**Closure Phase:** This is the last stage of STLC in which the process of testing is analysed.

**Scope and importance of Testing**:

Software **testing** contributes to determining or assess the product quality. The **scope** of software **testing** itself is to cover both functional and non-functional aspects of the entire product under development/**test**. The function requirements are the use cases relevant to the end user visible features of the product.



**SECOUND DAY TOPICS:**

**APPROACHES TO DEVELOPMENT:**

**TOP**-**DOWN DEVELOPMENT:**

**top**-**down development is** an **approach** to program **development** in which progress is made by defining required elements in terms of more basic elements, beginning with the required program and ending when the implementation language is reached

**BOTTOM UP APPROACH:**

A **bottom**-**up approach** is the piecing together of systems to give rise to more complex systems, thus making the original systems sub-systems of the emergent system. **Bottom**-**up** processing is a type of information processing based on incoming data from the environment to form a perception.

**WHITE BOX TESTING**

**White box testing*:***With interior knowledge of software

* Internal programming fully known.
* Tester has full knowledge of internal working of the application
* Known as glass, open box, clear box, structural testing or code-based testing.
* Performed by testers and developers.
* Internal working is fully known, and tester can design test data accordingly.
* Most exhaustive and time consuming.
* Data domain and internal boundaries can be better tested.
* Suited to algorithm testing

**Black box testing:**

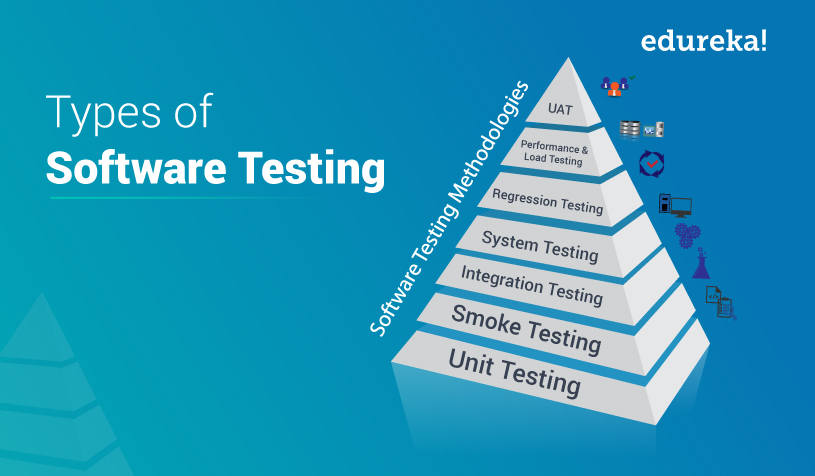
 Without having interior knowledge of software

* Internal programming not known.
* Internal workings of an application are not required to be known.
* Known as closed box, data driven and functional testing.
* Performed by end users and also by testers and developers.
* Testing is based on external expectation; internal behaviour of application is unknown.
* Least time consuming and exhaustive.
* Not suited to algorithm testing.
* It can be done by trial and error method.

**Gary box testing:**

Combination of ***white-box testing***and ***black-box testing***. The aim of this testing is to search for the defects if any due to improper structure or improper usage of applications. Internal programming partially known.

* Somewhat knowledge of internal working of application are known.
* Known as translucent testing.
* Performed by end users and and also by testers and developers.
* On the basis of high-level database diagrams and data flow diagram.
* Partly time consuming and exhaustive.
* Not suited to algorithm testing.



**PHASES OF TESTING**

**UNIT TESTING:**

testing of individual software component module is termed as unit testing.

it typically done by the programmers and not by the testers.

**SMOKE TESTING:**

**whenever** a new build is provided by the developers then the software testing team validates the build and ensures that no major issues exist.

**SANITY TESTING:**

**sanity testing** is done to determine if new software version is performing well enough to accept it for a major testing effort or not .

if the application is crashing for the initial use then the system is not stable enough for further testing.

**SYSTEM TESTING:**

**The** entire system is tested as per the requirements.

it is a black box type testing that is based on overall requirements specification and covers all the combined part of the system.

**INTEGRATION TESTING:**

**Testing** of all integrated modules to verify the combined functionality after integration is termed as integration testing.

**END-TO-END TESTING:**

End to end testing involves testing of a complete application environment in situation that mimics real world use, such as interacting with a database, using network communications.

**USER ACCEPTANCE TEST:**

**AN** acceptance testing is done by the client and verifies by the client and verifies whether the end to end flow of the system is as per the business requirement or not.

and if it is as per the needs of the end user, client accepts the software only when all the features and functionalities work as expected.

**RETESTING:**

**Te**sting the defect again after correction.

**REGRESSION TESTING:**

This testing is done for verification of whether the correction of one defect is caused any other defects or not.

this is the continuous test, cross checking

**ADHOC TESTING:**

**The** name itself suggests that this testing is performed on ad-hoc basis with no reference to the test cases and also without any plan or documentation.

**BROWSER COMPATIBILITY TESTING:**

it is a subtype of compatibility testing and it is performed by the testing team.

whether the browser is compatible with the code or not.

**STATE TRANSATION TESTING:**

**Up**dating the data from central hub to the webpage.

**PERFORMANCE TESTING:**

**It** is also a kind of automation testing for performance

it will increase the users gradually.

THIRD DAY TOPICS:

**TEST CASES**

A TEST CASE is a set of conditions or variables under which a tester will determine whether a system under test satisfies requirements or works correctly. The process of developing test cases can also help find problems in the requirements or design of an application

**DIFFRENCE BETWEEN SCENARIO AND TEST CASES**

**Test case** consist of a set of input values, execution precondition, expected results and executed post condition, developed to cover certain **test** condition. While **Test scenario** is nothing but a **test** procedure. A **Test Scenarios** has one or many relations with **Test Case**, meaning a **scenario** can have multiple **test cases**

**TEMPLATE OF TEST CASE:**

***REQUIREMENT ID:*** Unique id is required for each requirement which is helpful in identifying.

***TEST CASE ID:*** Unique ID is required for each test case. Follow some convention to indicate the types of the test. **For Example, ‘**TC\_UI\_1' indicating ‘user interface test case #1'.

***TEST CASE NAME:*** Test case title. **For Example,**verify the login page with a valid username and password.

***TEST CASE DESCRIPTION:*** Describe the test objective in brief.

***TEST STEP NO:*** Unique no which is required for each test step.

***TEST STEP DESCRIPTION:*** List all the test execution steps in detail. Write test steps in the order in which they should be executed. Make sure to provide as many details as you can.

***TEST DATA:*** Use of test data as an input for this test case. You can provide different data sets with exact values to be used as an input.

***EXPECTED VALUE:*** What should be the system output after test execution? Describe the expected result in detail including message/error that should be displayed on the screen.

***ACTUAL VALUE:*** The actual test result should be filled after test execution. Describe the system behaviour after test execution.

***PREREQUISITE:*** What would be the state of the system after running the test case?

***STATUS:*** If an actual result is not as per the expected result, then mark this test as **failed**. Otherwise, update it as **passed**.

***DEFECT ID:*** If the test status is **failed**, then include the link to the defect log or mention the defect number.

***TESTER:*** The person who is responsible to validate all the test cases.

***AUTOMATION:*** Whether this test case is automated or not. It is useful to track the automation status when test cases are automated.

**BOUNDARY VALUE TESTING:**

Boundary value analysis is a type of black box or specification-based testing technique in which tests are performed using the boundary values.

## **Example:**

An exam has a pass boundary at 50 percent, merit at 75 percent and distinction at 85 percent. The Valid Boundary values for this scenario will be as follows:

49, 50 - for pass

74, 75 - for merit

84, 85 - for distinction

Boundary values are validated against both the valid boundaries and invalid boundaries.

The Invalid Boundary Cases for the above example can be given as follows:

0 - for lower limit boundary value

101 - for upper limit boundary value

## **Equivalence Partitioning Testing:**

Equivalence Partitioning also called as equivalence class partitioning. It is abbreviated as ECP. It is a software testing technique that divides the input test data of the application under test into each partition at least once of equivalent data from which test cases can be derived.

An advantage of this approach is it reduces the time required for performing testing of a software due to a smaller number of test cases.

## **Example:**

The Below example best describes the equivalence class Partitioning:

Assume that the application accepts an integer in the range 100 to 999

Valid Equivalence Class partition: 100 to 999 inclusive.

Non-valid Equivalence Class partitions: less than 100, more than 999, decimal numbers and alphabets/non-numeric characters.

**SILLY DEFECTS:**

**These** defects are not having any severity and the defects are most commonly name errors.

**Severity:**

**Severity** is defined as the degree of impact a Defect has on the development or operation of a component application being tested. Higher effect on the system functionality will lead to the assignment of higher **severity** to the bug. Quality Assurance engineer usually determines the **severity** level of defect

## **Priority:**

## Priority is defined as the order in which a defect should be fixed. Higher the priority the sooner the defect should be resolved.

Defects that leave the software system unusable are given higher priority over defects that cause a small functionality of the software to fail.

## **Defect Severity and Priority Types**

In Software Testing, Defect severity can be categorized into four class

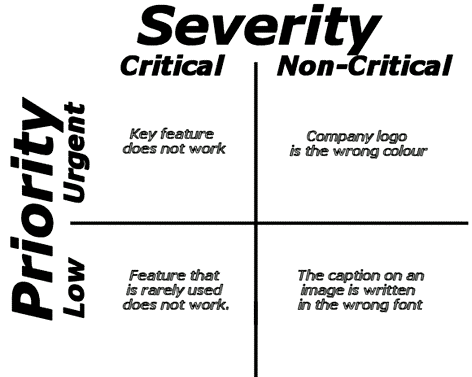
* **Critical**: This defect indicates complete shut-down of the process, nothing can proceed further
* **Major**: It is a highly severe defect and collapses the system. However, certain parts of the system remain functional
* **Medium**: It causes some undesirable behaviour, but the system is still functional
* **Low**: It won't cause any major break-down of the system

Defect priority can be categorized into three class

* **Low:**The Defect is an irritant, but repair can be done once the more serious Defect has been fixed
* **Medium:**During the normal course of the development activities defect should be resolved. It can wait until a new version is created
* **High:**The defect must be resolved as soon as possible as it affects the system severely and cannot be used until it is fixed

## **Tips for determining the Severity of a Defect**

* **Decide the frequency of occurrence:** In some cases, if the occurrence of a minor defect is frequent in the code, it can be more severe. So, from a user's perspective, it is more serious even though it is a minor defect.
* **Isolate the defect:** Isolating the defect can help to find out its severity of the impact.

[](https://www.guru99.com/images/5-2015/050115_0552_PriorityVsS1.gif)

**Defect Life Cycle:**

In this tutorial, I will talk about the life cycle of a defect to make you aware of the various stages of a defect which a tester has to deal with while working in a testing environment.

I have also added the most frequently asked interview questions on the Defect Life Cycle. This is important to know about the various states of a defect for understanding the life cycle of a defect. The main intention of performing a testing activity is to check if the product has any issues/errors.

In terms of real scenarios, errors/mistakes/faults are all referred to as bugs/defects and hence we can say, that the main objective of doing testing is to assure that the product is less prone to defects (no defects is an unrealistic situation).

**Defect Life Cycle in Detail:**

A Defect life cycle, also known as a Bug life cycle, is a cycle of a defect from which it goes through covering the different states in its entire life. This starts as soon as any new defect is found by a tester and comes to an end when a tester closes that defect assuring that it won’t get reproduced again.

**Defect Workflow**

A screenshot of a social media post

Description automatically generated

#### **Defect States**

***New****:*  This is the first state of a defect in the Defect Life Cycle. When any new defect is found, it falls in a ‘New’ state and validations and testing are performed on this defect in the later stages of the Defect Life Cycle.

***Assigned:*** In this stage, a newly created defect is assigned to the development team for working on the defect. This is assigned by the project lead or the manager of the testing team to a developer.

***Open:***Here, the developer starts the process of analyzing the defect and works on fixing it, if required. If the developer feels that the defect is not appropriate then it may get transferred to any of the below four states namely **Duplicate, Deferred, Rejected or Not a Bug**-based upon the specific reason.

I will discuss these four states in a while.

***Fixed:***When the developer finishes the task of fixing a defect by making the required changes then he can mark the status of the defect as ‘Fixed’.

***Pending Retest:***After fixing the defect, the developer assigns the defect to the tester for retesting the defect at their end and till the tester works on retesting the defect, the state of the defect remains in ‘Pending Retest’.

***Retest:***At this point, the tester starts the task of working on the retesting of the defect to verify if the defect is fixed accurately by the developer as per the requirements or not.

***Reopen:***If any issue still persists in the defect then it will be assigned to the developer again for testing and the status of the defect gets changed to ‘Reopen’.

***Verified:***If the tester does not find any issue in the defect after being assigned to the developer for retesting and he feels that if the defect has been fixed accurately then the status of the defect gets assigned to ‘Verified’.

***Closed:***When the defect does not exist any longer then the tester changes the status of the defect to ‘Closed’.

**Test Plan:**

A**TEST PLAN** is a document describing software testing scope and activities. It is the basis for formally testing any software/product in a project.

The test plan consists of the following elements:

**Overview of:** What is the overview of the project.

**In Scope / Out Scope**: This consists of the requirements which are in the scope and out of the scope.

**Schedule and estimation**: This include the plan according to particular time.

**Staff Planning**: This includes the total staff that are working for a particular project.

**Types of testing**: The types of testing that is required for the project is included in it.

**Risks and mitigation plan**: The risks involved in it are included.

**Entry and Exit Criteria:**

**Entry:** Entry criteria for testing can be defined as “Specific conditions or on-going activities that must be present before a process can begin.”

* The requirement document should be available.
* Complete understanding of the application flow is required.
* The Test Plan Document should be ready.

**Exit**: “The specific conditions or on-going activities that should be fulfilled before completing the software testing life cycle.

* Test Cases should be written and reviewed.
* Test Data should be identified and ready.
* Test automation script should be ready if applicable.

**Suspension Criteria:**

**Suspension criteria** & resumption **requirements**. **Suspension criteria** specify the **criteria** to be used to **suspend** all or a portion of the testing activities while resumption **criteria** specify when testing can resume after it has been **suspended**. When a defect is introduced that cannot allow any further testing.

**Automation Scope:** Automation is the process of evaluating the AUT(Application under Test) against the specification with the help of a tool.

Depending on the nature of testing there are two main branches under automation.

1. Functional testing with automation.
2. Performance testing with automation.

Functional Testing with the help of Automation.

Functional automated testing has emerged as a key area in most of the testing processes. The main area where the functional testing tools are used is for regression test case execution. Mostly in the agile scrum methodology where frequent releases are happening, it is almost impossible to execute all the regression test cases manually with the short span of time. Automation gives a high ROI(Return of Investment) in this area since it is a one time effort for generating the scripts..

Performance Testing with the help of Automation.

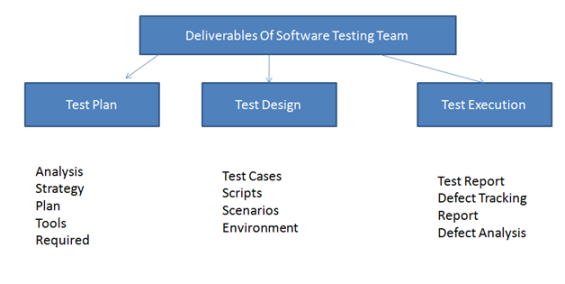
Performance testing is the process of evaluating the application performance which is being a critical requirement of every application now a days. Performance testing is almost impossible by manual means. There are different tools used across organizations for evaluating application performance.

**Test Environment Setup:** Test Environment consists of elements that support test execution with software, hardware and network configured.

* It is a combination of hardware and software environment on which the tests will be executed.
* It includes hardware configuration, operating system settings, software configuration, test terminals and other support to perform the test.
* It is the most crucial aspect of the testing process. Unavailability or faulty environment setup can ruin all the testing efforts.
* Practically, the QA team cannot start actual work without having the right environment to test.
* Readiness of the test environment can be validated by smoke testing, and performed by the QA team.
* Ideally, we can say that the Entry Criteria of this phase is the provision of Test Plan, readiness of Smoke Test cases and preparation of test data.
* The exit criteria of this phase is that the test environment should be ready and smoke testing should be performed successfully with expected results.
* Activites that are performed in this phase like design test environment ,set up test environment.

**Deliverables:**

Test Deliverables are documents that are given to the stakeholders when the software is being developed. In this section we will discuss the following test deliverables:

  
**Deliverables Of Software Testing Team**