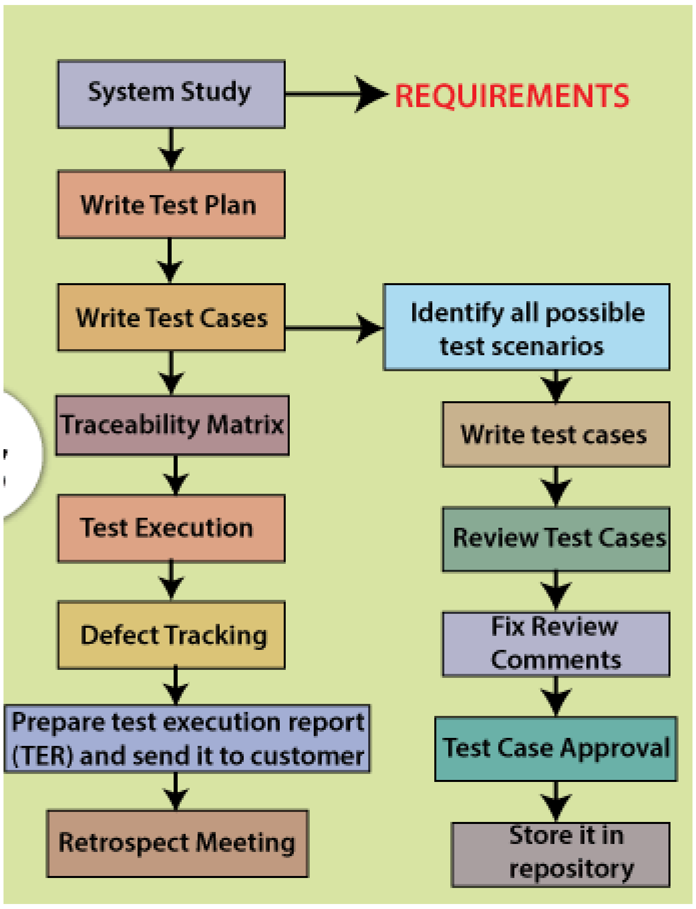
STLC- Software Testing Life Cycle



STLC stands for Software Testing Life Cycle, and it represents the series of activities performed during the testing process of a software application or system.

System Study: Customer give requirements in the form of CRS. BA will convert CRS to SRS and BA will explain how each and every feature works for the entire team. BA will also explain customer business scenarios to teammates. Every Test Engineer will understand requirements, while understanding requirements, if they find any issues, they will talk to BA and clarify the doubts. This complete process of understanding the requirements is called System study.

Test Plan: It is a document which derives all the future testing activities. Test plan document consists of information regarding how TE are required to test in future, which features are to be tested by which Test Engineers, what types of testing to be done on project in future, roles and responsibilities of every Engineer working on project, which Test cases management tool to use, which defect tracking tool to use in future will be maintained in Test plan document.

Test case: Once the Test plan is completed, every TE will start writing Test cases from their assigned modules, where the TE will follow the procedure to write the Test cases. First do System study for assigned modules, later identify all the possible scenarios, document it and conduct brainstorming meetings to have good coverage, later once scenarios are ready, TE will write Test cases by applying Test case design Technique in Test case Template and get it reviewed by reviewers. If there is any problem in Test cases, it will be fixed by TE and later it will be approved by Test Lead. Finally Test case will be Test case repository.

Prepare Traceability matrix – It is a document which ensures that each and every requirement has got at least 1 Test case. There are 3 types of Traceability matrix. They are:

1) **Forward Traceability Matrix** – It is used to map requirements to the test cases. This is done before Test case execution. Here we are ensuring that the product developments are going in the right direction.

2) **Backward Traceability Matrix** – It is used to map Test cases to the requirement. This is done after Test case execution. Here we are ensuring that we are not going against/developing products against customer requirements.

3) **Bi-directional Traceability Matrix** – It is a combination of both Forward Traceability Matrix and Backward Traceability Matrix.

Advantages of RTM:

1) Ensures complete requirements are documented.

2) It helps in analyzing the root cause of any defect.

3) Applications can be developed according to the requirement.

Test case execution: In this stage, developers will give the build to Test Engineers and Test Engineers will start to test the build by executing the Test cases against the application where in TE will do Smoke testing, Functional Testing, Integration Testing, System Testing etc. Since Test Engineers are testing the application by looking into Test cases, it is called Test case execution. Here, we follow the procedure to execute the test cases.

Defect Tracking: While executing the test cases, if the Test engineer finds any defects, he will login to Defect tracking tool and prepare defect report and communicate to developer with unique defect id. Developer will fix the defect and TE will retest the defect and if the defect is really fixed. He will close the bug or TE will re-open the bug. This is called Defect Tracking.

Test case execution report: TL's will prepare Test case execution report/test case summary report for every build/feature/test cycle/ end of every Release. This report consists of how Test cases are written, executed, not executed, pass or fail. This report will be sent to Management team, Development team, Testing team, customers

Release Retrospect meeting or Project closure meeting: Once Testing is completed, at the end of the Project, Test manager will do this meeting called as Release Retrospect meeting where in he will invite all the Test engineers who has worked on project and discuss about the mistakes and achievements done by Test engineers, while working on project.

Challenges in Software Testing:

1) **Complexity of Software**: As software systems become more complex, testing becomes more challenging. Interactions between different components, integration issues, and the need for testing across various platforms and devices add to the complexity.

2) **Changing Technology**: Rapid advancements in technology and the adoption of new tools and frameworks can make it challenging for testing teams to keep up. Learning and implementing new testing tools can be time-consuming.

3) **Regression Testing**: As software evolves, changes to one part of the code can unintentionally introduce defects in other areas. Regression testing is essential to ensure that new features or bug fixes do not negatively impact existing functionalities. However, running comprehensive regression tests can be time-consuming.

4) **Incomplete Requirements:** Testing without clear and complete requirements can lead to ambiguity in test cases. Incomplete or changing requirements make it difficult to create comprehensive test cases, leading to potential issues being overlooked.

5) **Time Constraints**: Often, software development projects are subject to tight deadlines. Testing is sometimes compressed, leaving limited time for thorough testing. This can result in inadequate test coverage and potentially missed defects.

6) **Communication Issues**: Effective communication between development and testing teams is essential. Miscommunication or a lack of collaboration can result in misunderstandings about requirements, leading to ineffective testing.

Principles of Software Testing

1) We should not do Exhaustive Testing
2) We should do early Testing
3) Testing should be done to identify the bugs in the software
4) We should focus on “Pesticide paradox”

5) We should focus on “Defect clustering”
6) Testing is context dependant(depending on the type of application and customer requirements, testing should be done)
7) Absence of errors in the software does not mean that software is free from defects.

Pesticide Paradox: If the same test cases are run for more number of executions, then the test case might not have capability to catch new bugs, hence we need to re-update our test cases. This concept is called the Pesticide Paradox.

Defect clustering: Non-uniform distribution of defects across the features or one feature having more number of bugs, another feature having very less number of defects can be termed as Defect clustering.

Difference between Defect, Bug, Error and Failure

1) **Defect** – When developers are doing Unit testing to check whether functionality is perfect or not in their local set up, now they compare actual result with the expected result, there is any difference between the actual result and expected result, this leads to a defect. Basically this occurred during the development phase.

2) **Bug** – When Tester is testing the application, if they find variation between actual result and expected result, this will be considered as Bug. Also defects accepted by developers can be considered as bugs. Bug is also an informal name given to the defect.

3) **Error** – Mistakes done in the program while writing the code, because of which the developer is not able to write and compile the code is called an Error.

4) **Failure** – Once software is developed, it is tested and verified by our testers. Finally end users are facing issues in the production, it is termed as failure.

Static and Dynamic Testing

Static Testing – Testing the project related documents in the form of Review, walkthrough and Inspection is called Static Testing.

Review – To check if the document is correct and complete.

Review can be done by single person

Reviews can be

a) Requirement Review

b) Design Review

d) Test Plan Review

WalkThrough – This needs to be done by multiple people.

a) It is a Informal review

b) Author reads the document and discuss with the peers

c) It is not planned and can be done anytime

d) No time limits

Inspection :

1) It is a formal review type

2) 3-8 people will be part of this meeting. Readers (incharge of the document), writers(on who writes the questions), moderator(organizer) will be involved

3) Inspection is a proper schedule and will be informed to developer and Tester according to the schedule

Dynamic Testing – Testing the actual software by Unit Testing, Integration Testing, System Testing, User Acceptance Testing.

Why should we write Test cases?

1. To have better test case coverage – Once a customer gives a requirement, the developer will be busy in writing the code, TE at this point of time should start writing test cases and keep it ready. So we will have better Test case coverage while testing the software.

2. To have better consistency in Testing – If we already have our test cases ready, we can test the given build by looking into it, even if developer makes any changes in future build and it affects old feature, we can look into old test cases and start testing the software, so we will have better consistency in Testing.

3. If any new TE joins the company, he can look into the test case document and start testing the application. By this we can save a lot of time on training and can utilize resources efficiently.

4. Test cases are the only proof to show to developers and customers that we have covered all possible scenarios and tested the software.

5. We will not miss any scenarios and defects if we write Test cases.

6. Testing can be done in an organized way, if test cases are written.

Test case review process

Test case review, is a systematic process of examining and evaluating test cases to ensure their quality, accuracy, and effectiveness. The goal of test case review is to identify and correct defects or deficiencies in the test cases before they are executed, ultimately contributing to the improvement of the overall testing process. This review process involves collaboration among team members, including testers, developers, and Test Leads.

Key objectives of test case review include:

Ensuring Accuracy – Reviewing Test cases ensures test cases are in alignment with the project requirements.

Identifying Ambiguities– Reviewing Test cases helps us to know if we have missed any test scenario, test cases or written wrong test cases or repeated test cases.

Verification of Coverage – Reviewing test cases ensures that test cases cover all the relevant functionalities within the software.

Consistency Checks – Reviewing test cases ensures consistency is maintained which makes test cases potentially capable of catching bugs.

Feedback and Collaboration – Reviewing test cases provides an opportunity for collaboration among the team members to share their insights and contribute to the improvement of Test cases.

Process of conducting Test case Review:

1) Prepare relevant test case documents
2) Get the test cases reviewed by Test Leads, other Testers or who is having good product knowledge
3) Once review is conducted – Check for the discrepancies, duplicate test cases or missing test cases
4) Ensure feedback is documented and correct the Test cases accordingly
5) Update the Test cases accordingly and ensure that the updated test cases align with the Project objectives.

Hence, we can conclude that Test case reviews contribute to the overall quality of the testing process and the reliability of test results.