STLC & 7 PRINCIPLES OF TESTING

Sumia Akter Ria

Introduction of STLC

KNOW THE STLC

Introduction of STLC

Software Testing Life Cycle (STLC) is a process used to test software and ensure that quality standards are met.

Software Testing Life Cycle

Requirement Analysis

- In this phase, QA engineers gather and understand the software requirements.
- This includes functional requirements, non-functional requirements, and user stories.
- Requirements can be gathered from various sources, such as requirements documents, user stories, and system design documents.

Test Planning

- QA engineers create a test plan that outlines the scope of testing, the testing strategy, and the resources that will be required.
- The test plan should also include the schedule for testing and the risk assessment.

Test Case Development

- QA engineers develop test cases to test all software requirements.
- Test cases should be clear, concise, and easy to follow.
- They should also include the expected results for each test case.

Environment Setup

- In this phase, QA engineers set up the test environment.
- The test environment should be a replica of the production environment as much as possible.
- This includes the hardware, software, and data that will be used for testing.

Test Execution

- In this phase, QA engineers execute the test cases that were designed in the previous phase.
- They will record the actual results of each test case and compare them to the expected results.
- Any discrepancies between the actual and expected results are documented as defects.

Test Cycle Closure

- In this phase, QA engineers report on the results of testing.
- The test report should include the number of test cases executed, the number of defects found, and the severity of the defects.

PRINCIPLE OF TESTING

KNOW THE PRINCIPLES

TESTING SHOWS THE PRESENCE OF DEFECTS

- Testing exposes defects, not the absence of them.
- Testing increases confidence, not guarantees.
- Zero defects in testing don’t equal zero defects overall.

EXHAUSTIVE TESTING IS IMPOSSIBLE

- Tests software functionality with all possible inputs and pre-conditions.
- Impractical due to the vast number of test cases.
- Limited testing to some cases, assuming correctness.

EARLY TESTING SAVES TIME AND MONEY

- Identify defects during requirement analysis, reducing fixing costs.
- Catch bugs early in the SDLC, preventing rework in later stages.
- Ensure requirements are clear and well-defined before development begins.

DEFECTS CLUSTER TOGETHER

- Bugs are not evenly distributed; they cluster in a few modules.
- 80% of issues come from 20% of modules. Focus testing there!

BEWARE OF THE PESTICIDE PARADOX

- Using the same tests stops finding new bugs (like insects developing immunity to pesticides).
- Regularly review and update test cases to find more defects.
- Not finding new bugs ≠ bug-free software.

TESTING IS CONTEXT DEPENDENT

- Testing approach varies based on the software's purpose.
- Each application has specific requirements driving the testing strategy.
- Different methodologies, techniques, and testing types are used based on the application's nature.

ABSENCE-OF-ERRORS IS A FALLACY

- Some organizations expect that testers can run all possible tests and find all possible defects, but principles 2 and 1, respectively, tell us that this is impossible.
- Software must fulfill all customer requirements to be usable.
- Being 99% bug-free isn't sufficient if it doesn't meet user needs. Fulfilling user requirements is mandatory for software usability.

THANK YOU 

Sumia Akter Ria