Unit 4 Notes

Software Testing Life Cycle (STLC)

The Software Testing Life Cycle (STLC) is a systematic approach to testing a software application to ensure that it meets the requirements and is free of defects. It is a process that follows a series of steps or phases, and each phase has specific objectives and deliverables. The STLC is used to ensure that the software is of high quality, reliable, and meets the needs of the end-users.

The main goal of the STLC is to identify and document any defects or issues in the software application as early as possible in the development process. This allows for issues to be addressed and resolved before the software is released to the public.

Phases of STLC

1. Requirement Analysis: Requirement Analysis is the first step of the Software Testing Life Cycle (STLC). In this phase quality assurance team understands the requirements like what is to be tested. If anything is missing or not understandable then the quality assurance team meets with the stakeholders to better understand the detailed knowledge of requirements.

2. Test Planning: Test Planning is the most efficient phase of the software testing life cycle where all testing plans are defined. In this phase manager of the testing, team calculates the estimated effort and cost for the testing work. This phase gets started once the requirement-gathering phase is completed.

The activities that take place during the Test Planning stage include:

- Identifying the testing objectives and scope
- Developing a test strategy: selecting the testing methods and techniques that will be used
- Identifying the testing environment and resources needed
- Identifying the test cases that will be executed and the test data that will be used
- Estimating the time and cost required for testing
- Identifying the test deliverables and milestones
- Assigning roles and responsibilities to the testing team
- Reviewing and approving the test plan

3. Test Case Development: The test case development phase gets started once the test planning phase is completed. In this phase testing team notes down the detailed test cases. The testing team also prepares the required test data for the testing. When the test cases are prepared then they are reviewed by the quality assurance team.

4. Test Environment Setup: Test environment setup is a vital part of the STLC. Basically, the test environment decides the conditions on which software is tested. This is independent activity and can be started along with test case development. In this process, the testing team is not involved. either the developer or the customer creates the testing environment.

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

The activities that take place during the test execution stage of the Software Testing Life Cycle (STLC) include:

- **Test execution:** The test cases and scripts created in the test design stage are run against the software application to identify any defects or issues.
- **Defect logging:** Any defects or issues that are found during test execution are logged in a defect tracking system, along with details such as the severity, priority, and description of the issue.
- **Test data preparation:** Test data is prepared and loaded into the system for test execution
- **Test environment setup:** The necessary hardware, software, and network configurations are set up for test execution
- **Test execution:** The test cases and scripts are run, and the results are collected and analyzed.
- **Test result analysis:** The results of the test execution are analyzed to determine the software's performance and identify any defects or issues.
- **Defect retesting:** Any defects that are identified during test execution are retested to ensure that they have been fixed correctly.
- **Test Reporting:** Test results are documented and reported to the relevant stakeholders.

It is important to note that test execution is an iterative process and may need to be repeated multiple times until all identified defects are fixed and the software is deemed fit for release.

6. Test Closure: Test closure is the final stage of the Software Testing Life Cycle (STLC) where all testing-related activities are completed and documented. The main objective of the test closure stage is to ensure that all testing-related activities have been completed and that the software is ready for release.

At the end of the test closure stage, the testing team should have a clear understanding of the software's quality and reliability, and any defects or issues that were identified during testing should have been resolved. The test closure stage also includes documenting the testing process and any lessons learned so that they can be used to improve future testing processes.

Test plan

- In software testing, a test plan gives detailed testing information regarding an upcoming testing effort, including Scope of testing, Schedule, Test Deliverables, Release Criteria, Risks and Contingencies.

Test cases

Test cases are documents that contain a set of test data, preconditions, expected results, actual results, and post conditions which are developed to verify compliance against a specific requirement.

1. **Design the Test Case for following:**

    - Pen

Sno	Test Condition	Expected Result
1	Verify the company logo on the pen.	Logo
2.	Verify the dimensions of the pen.	Correct Dimensions should be there as given in the requirement specifications.
3.	Verify the material of the pen	Material should be plastic with a rubber grip.

4.	Verify the grip of the pen	Grip should be hold and work on
5.	Verify all the parts of the pen	All the parts i.e. the body , cap , refill should be present.
6.	Verify that the pen is smooth in writing.	Go glitches should be there when writing with a pen.
6.	Verify that the color of the ink and the cap matches	Cap and the ink color should be the same.
7.	Verify that the pen is working for all the angles .	Pen should work fine for all the writing angles.
8.	Verify the tolerance on the nib by putting additional pressure.	Pen nib should not break down.
9.	Verify how much does the pen writes	Should write for the specific distance in m as specified in the requirements specification.
10.	Verify the shape of the pen.	Should be appropriate for proper writing.

Test Scenario

A Test Scenario is defined as any functionality that can be tested. It is also called Test Condition or Test Possibility. As a tester, you should put yourself in the end user's shoes and figure out the real-world scenarios and use cases of the Application Under Test.

Test Automation & Testing Tools

Automation testing tools allow you to effortlessly create, run, and maintain tests and support a centralized view of analytics of test results.

Product	Katalon	Selenium	appium	TestComplete	cypress
Application Under Test	Web/API/ Mobile/Desktop	Web	Mobile (Android/iOS)	Web/Mobile/ Desktop	Web
Supported platform(s)	Windows/ macOS/ Linux	Windows/ macOS/ Linux/Solaris	Windows/ macOS	Windows	Windows/ macOS/ Linux
Setup & configuration	Easy	Coding Required	Coding Required	Easy	Coding Required
Low-code & Scripting mode	Both	Scripting Only	Scripting Only	Both	Scripting Only
Supported language(s)	Java & Groovy	Java, C#, Python, JavaScript, Ruby, PHP, Perl	Java, C#, Python, JavaScript, Ruby, PHP, Perl	JavaScript, Python, VBScript, JScript, Delphi, C++, C#	JavaScript
Advanced test reporting	✓	✗	✗	✗	✓
Pricing	Free and Paid	Free	Free	Paid	Free and Paid
Ratings & Reviews (Gartner)	4.4/5 740 reviews	4.5/5 443 reviews	4.4/5 90 reviews	4.4/5 45 reviews	4.6/5 27 reviews

Manual Testing Tools

1. LoadRunner
2. Citrus
3. ZAP
4. NUnit
5. JIRA
6. SonarQube
7. JMeter
8. Bugzilla
9. Mantis
10. Tessy
11. TestLink

Difference Between Manual Testing and Automated Testing

Parameters	Manual Testing	Automation Testing
Definition	In manual testing, the test cases are executed by the human tester.	In automated testing, the test cases are executed by the software tools.
Processing Time	Manual testing is time-consuming.	Automation testing is faster than manual testing.
Resources requirement	Manual testing takes up human resources.	Automation testing takes up automation tools and trained employees.
Exploratory testing	Exploratory testing is possible in manual testing.	Exploratory testing is not possible in automation testing.
Reliability	Manual testing is not reliable due to the possibility of manual errors.	Automated testing is more reliable due to the use of automated tools and scripts.
Investment	In manual testing, investment is required for human resources.	In automated testing, investment is required for tools and automated engineers.
Test results availability	In manual testing, the test results are recorded in an excel sheet so they are not readily available.	In automated testing, the test results are readily available to all the stakeholders in the dashboard of the automated

Parameters	Manual Testing	Automation Testing
		tool.
Human Intervention	Manual testing allows human observation, thus it is useful in developing user-friendly systems.	Automated testing is conducted by automated tools and scripts so it does not involve assurance of user-friendliness.
Programming knowledge	There is no need for programming knowledge in manual testing.	Programming knowledge is a must in case of automation testing as using tools requires trained staff.
When to Use?	Manual testing is usable for Exploratory testing, Usability testing, and Adhoc testing.	Automated testing is suitable for Regression testing, Load testing, and Performance testing

Characteristics of modern tools in software testing are:

1. They have an improved speed.
2. They are more efficient and accurate.
3. They are easy to use.
4. They can be automated.

Agile methodology

The Agile methodology is a project management and software development approach that emphasizes flexibility, collaboration, and customer-centricity. It is the latest model used by major companies today like Facebook, google, amazon, etc. It follows the iterative as well as incremental approach that emphasizes the importance of delivering of working product very quickly.

The Four Values of The Agile Manifesto

1. Individuals and Interactions Over Processes and Tools

2. Working Software Over Comprehensive Documentation

3. Customer Collaboration Over Contract Negotiation

4. Responding to Change Over Following a Plan

The Twelve Agile Manifesto Principles

1. **Customer satisfaction through early and continuous software delivery –**
2. **Accommodate changing requirements throughout the development process –**
3. **Frequent delivery of working software –**
4. **Collaboration between the business stakeholders and developers throughout the project –**
5. **Support, trust, and motivate the people involved –**
6. **Enable face-to-face interactions –**
7. **Working software is the primary measure of progress –**
8. **Agile processes to support a consistent development pace –**
9. **Attention to technical detail and design enhances agility –**
10. **Simplicity –**
11. **Self-organizing teams encourage great architectures, requirements, and designs –**
12. **Regular reflections on how to become more effective –**

Challenges in Adopting Agile Method.

- Resistance To Change. ...
- Lack Of Clarity
- Cultural change

What are Agile frameworks?

Agile frameworks are methods of organizing and dealing with software program development initiatives that follow the principles and values of the Agile Manifesto.

Extreme Programming (XP)

-

Extreme programming (XP) is one of the most important software development frameworks of Agile models. It is used to improve software quality and responsiveness to customer requirements.

Basic principles of Extreme programming

XP is based on the frequent iteration through which the developers implement User Stories. User stories are simple and informal statements of the customer about the functionalities needed. A User Story is a conventional description by the user of a feature of the required system. It does not mention finer details such as the different scenarios that can occur. Some of the basic activities that are followed during software development by using the XP model are given below:

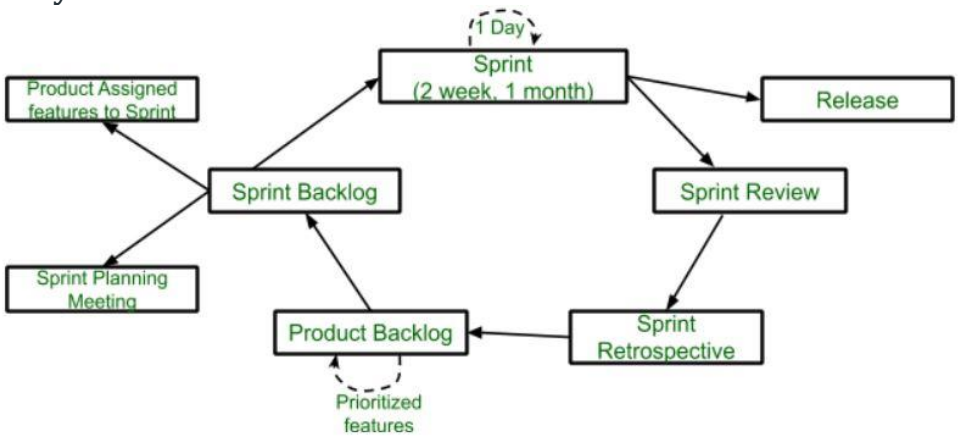
- **Coding**
- **Testing**
- **Listening**
- **Designing**
- **Feedback**
- **Simplicity**
- **Pair Programming**
- **Continuous Integration**
- **Refactoring**
- **Collective Code Ownership**

- **Planning Game**
- **On-site Customer**

Scrum (software development)

Scrum is the type of **Agile framework**. It is a framework within which people can address complex adaptive problem while productivity and creativity of delivering product is at highest possible values. Scrum uses **Iterative process**.

Lifecycle of Scrum:



Sprint: A Sprint is a time box of one month or less. A new Sprint starts immediately after the completion of the previous Sprint. **Release:** When the product is completed, it goes to the Release stage. **Sprint Review:** If the product still has some non-achievable features, it will be checked in this stage and then passed to the Sprint Retrospective stage. **Sprint Retrospective:** In this stage quality or status of the product is checked. **Product Backlog:** According to the prioritize features the product is organized. **Sprint Backlog:** Sprint Backlog is divided into two parts Product assigned features to sprint and Sprint planning meeting.

Advantage of using Scrum framework:

- Scrum framework is fast moving and money efficient.
- Scrum framework works by dividing the large product into small sub-products. It's like a divide and conquer strategy
- In Scrum customer satisfaction is very important.
- As Scrum framework rely on constant feedback therefore the quality of product increases in less amount of time

Disadvantage of using Scrum framework:

- Scrum framework do not allow changes into their sprint.
- The daily Scrum meetings and frequent reviews require substantial resources.

Software Reliability

Software reliability is also defined as the probability that a software system fulfills its assigned task in a given environment for a predefined number of input cases, assuming that the hardware and the input are free of error.

Software Failure Mechanisms

The software failure can be classified as:

Transient failure: These failures only occur with specific inputs.

Permanent failure: This failure appears on all inputs.

Recoverable failure: System can recover without operator help.

Unrecoverable failure: System can recover with operator help only.

Non-corruption failure: Failure does not corrupt system state or data.

Corrupting failure: It damages the system state or data.

Software failures may be due to bugs, ambiguities, oversights or misinterpretation of the specification that the software is supposed to satisfy, carelessness or incompetence in writing code, inadequate testing, incorrect or unexpected usage of the software or other unforeseen problems.

Reliability Metrics

Reliability metrics are used to quantitatively expressed the reliability of the software product. The option of which metric is to be used depends upon the type of system to which it applies & the requirements of the application domain.

Some reliability metrics which can be used to quantify the reliability of the software product are as follows:

1. Mean Time to Failure (MTTF)

MTTF is described as the time interval between the two successive failures

2. Mean Time to Repair (MTTR)

Once failure occurs, some-time is required to fix the error. **MTTR** measures the average time it takes to track the errors causing the failure and to fix them.

3. Mean Time Between Failure (MTBR)

We can merge **MTTF** & **MTTR** metrics to get the MTBF metric.

MTBF = MTTF + MTTR

4. Rate of occurrence of failure (ROCOF)

It is the number of failures appearing in a unit time interval. The number of unexpected events over a specific time of operation.

5. Probability of Failure on Demand (POFOD)

POFOD is the possibility that the system will fail when a service request is made.