Software Testing

Unit-2:

1. **The Test Development Life Cycle (TDLC)**
   Links - https://www.exforsys.com/tutorials/testing/life-cycle-of-testing-process.html

   Life Cycle of Testing Process
   This article explains about Differant steps in Life Cycle of Testing Process. in Each phase of the development process will have a specific input and a specific output. Once the project is confirmed to start, the phases of the development of project can be divided into the following phases:

- Software requirements phase.
- Software Design
- Implementation
- Testing
- Maintenance

   In the whole development process, testing consumes highest amount of time. But most of the developers oversee that and testing phase is generally neglected. As a consequence, erroneous software is released. The testing team should be involved right from the requirements stage itself.
   The various phases involved in testing, with regard to the software development life cycle are:

   1. Requirements stage
   2. Test Plan
   3. Test Design.
   4. Design Reviews
   5. Code Reviews
   6. Test Cases preparation.
   7. Test Execution
   8. Test Reports.
   9. Bugs Reporting
   10. Reworking on patches.
   11. Release to production.
   Requirements Stage
   Normally in many companies, developers itself take part in the requirements stage. Especially for product-based companies, a tester should also be involved in this stage. Since a tester thinks from the user side whereas a developer can't. A separate panel should be formed for each module comprising a developer, a tester and a user. Panel meetings should be scheduled in order to gather

everyone's view. All the requirements should be documented properly for further use and this document is called "Software Requirements Specifications".

Test Plan

Without a good plan, no work is a success. A successful work always contains a good plan. The testing process of software should also require good plan. Test plan document is the most important document that brings in a process – oriented approach. A test plan document should be prepared after the requirements of the project are confirmed. The test plan document must consist of the following information:

- Total number of features to be tested.
- Testing approaches to be followed.
- The testing methodologies
- Number of man-hours required.
- Resources required for the whole testing process.
- The testing tools that are to be used.
- The test cases, etc

Test Design

Test Design is done based on the requirements of the project. Test has to be designed based on whether manual or automated testing is done. For automation testing, the different paths for testing are to be identified first. An end to end checklist has to be prepared covering all the features of the project.

The test design is represented pictographically. The test design involves various stages. These stages can be summarized as follows:

- The different modules of the software are identified first.
- Next, the paths connecting all the modules are identified.

Then the design is drawn. The test design is the most critical one, which decides the test case preparation. So the test design assesses the quality of testing process.

Test Cases Preparation

Test cases should be prepared based on the following scenarios:

- Positive scenarios
- Negative scenarios
- Boundary conditions and
- Real World scenarios

<p}< p="">

Design Reviews

The software design is done in systematical manner or using the UML language. The tester can do the reviews over the design and can suggest the ideas and the modifications needed.

Code Reviews
Code reviews are similar to unit testing. Once the code is ready for release, the tester should be ready to do unit testing for the code. He must be ready with his own unit test cases. Though a developer does the unit testing, a tester must also do it. The developers may oversee some of the minute mistakes in the code, which a tester may find out.
Test Execution and Bugs Reporting
Once the unit testing is completed and the code is released to QA, the functional testing is done. A top-level testing is done at the beginning of the testing to find out the top-level failures. If any top-level failures occur, the bugs should be reported to the developer immediately to get the required workaround.

The test reports should be documented properly and the bugs have to be reported to the developer after the testing is completed.
Release to Production
Once the bugs are fixed, another release is given to the QA with the modified changes. Regression testing is executed. Once the QA assures the software, the software is released to production. Before releasing to production, another round of top-level testing is done.

The testing process is an iterative process. Once the bugs are fixed, the testing has to be done repeatedly. Thus the testing process is an unending process.
</p}<>

2. **When should testing stop ?**
   Link - https://momentumsuite.com/software-testing/when-to-stop-software-testing/

   Software testing is one of the most challenging and important responsibilities in the software development cycle. It is a process used to help identify the correctness, completeness, and quality of developed computer software. Rather than limiting this to a formally technical process used to investigate and measure the quality of the developed computer software in terms of correctness, completeness, and security, it is imperative to test the software components and the software as a whole for integrity, capability, reliability, efficiency, portability, maintainability, compatibility, and usability of the software.
   As a software tester, you can not test everything, but you should test enough to have confidence in your product being reliably delivered. Testers are responsible for identifying defects in the product, determining if they are valid and could be a showstopper or not. They create test plans and execute them to ensure that the software under development is ready for release. They establish verification criteria, maintain documentation, keep track of bugs and provide suggestions on

improving the product. The general objective of software testing is to improve quality and reliability by finding software errors before the product is released.

Software testing is a never-ending process

Testing is a never-ending process in a positive way. Testing still continues through the software development process and beyond, even after the product has been delivered to market. It isn't enough to have done your own testing on your work and left it at that! As you continue supporting and developing your software, new ideas and enhancements will arise which could affect any number of things within your original product. So don't be surprised if you find yourself going back over old code again.

So, as a software tester, you should be curious and you should focus on the process in an alert mode. Software testing is a critical part of the development process. It is important to pay attention to finer details and errors in the product of the application, with the aim of finding out why doesn't the product work as intended. It is not possible to find all the bugs that may exist, but we can stop them from releasing a product before it's ready. The question is, can we find these bugs prior to the release, and what is their severity?

So, when to stop software testing?

There are a number of important questions that one has to ask themselves when trying to figure out when is the right time to stop testing. But perhaps the most important one is "Do we have test coverage?" Testing an application can be very complicated. It takes a lot of resources and time that could be better spent elsewhere. To make sure you are testing effectively and efficiently, you should ask yourself: "Do we have adequate test coverage?"

This can be difficult to determine. Many modern software applications are so complex and run in such as interdependent environment, that complete testing can never be done. "When to stop testing" is one of the most difficult questions for a test engineer. Common factors in deciding when to stop are:

- Deadlines (release deadlines, testing deadlines.)
- Test cases completed with certain percentages passed
- Test budget depleted
- Coverage of code/functionality/requirements reaches
- A specified point
- The rate at which bugs can be found is too small
- Beta or Alpha Testing period ends
- The risk in the project is under an acceptable limit

The decision of stopping software testing

The decision of stopping testing is based on the level of the risk acceptable to the management. As testing is a never-ending process, it can never be assumed that 100% testing has been done, only minimizing the risk of shipping the product to a client that was tested X%. The risk can be measured by risk analysis, but for small duration / low budget / low resources projects, it can be deduced simply:

- Measuring Test Coverage.

- The number of test cycles.
- The number of high-priority bugs.
  With Momentum Suite, you can manage all your software testing processes. If you think you should stop your software testing process, you can control it from your favorite browser in seconds. You can also notify your team as an alert via popular tools such as Jira and Slack by integrating them into your Momentum Suite.

3. **Verification Strategies, Review, Walkthrough, Inspection Testing types and Techniquies**
   Link - https://www.geeksforgeeks.org/verification-methods-in-software-verification/

   Reviewing any software to find faults is known as Software Verification. Verification is the process of checking that software achieves its goal without any bugs. It is the process to ensure whether the product that is developed is right or not. The reviewing of a document can be done from the first phase of software development i.e. software requirement and analysis phase where the end product is the SRS document. There are many methods for practicing the verification of the software like peer-reviews, walkthroughs, inspections, etc. that can help us in the prevention of potential faults otherwise, it may lead to the failure of the software.
   Methods of Verification
   Here are some of the common methods that are required for Verification are listed below.
- Peer Reviews
- Walk Through
- Inspections
  Peer Reviews
  The easiest method and most informal way of reviewing the documents or the programs/software to find out the faults during the verification process is the Peer-Review method. In this method, we give the document or software programs to others and ask them to review those documents or software programs where we expect their views about the quality of our product and also expect them to find the faults in the program/document. The activities that are involved in this method may include SRS document verification, SDD verification, and program verification. In this method, the reviewers may also prepare a short report on their observations or findings, etc.
  Advantages of Peer Reviews:
- You can expect some good results without spending any significant resources.
- It is very efficient and significant in its nature.
  Disadvantages of Peer Reviews:
- Lead to bad results if the reviewer doesn't have sufficient knowledge.
  Walk-Through

Walk-throughs are a formal and very systematic type of verification method as compared to peer review. In a walkthrough, the author of the software document presents the document to other persons which can range from 2 to 7. Participants are not expected to prepare anything. The presenter is responsible for preparing the meeting. The document(s) is/are distributed to all participants. At the time of the meeting of the walk-through, the author introduces the content in order to make them familiar with it and all the participants are free to ask their doubts.

Advantages of Walk-Through

- It may help us to find potential faults.
- It may also be used for sharing documents with others.

Disadvantages of Walk-Through

- The author may hide some critical areas and unnecessarily emphasize some specific areas of his / her interest.

Inspections

Inspections are the most structured and formal type of verification method and are commonly known as inspections. A team of three to six participants is constituted which is led by an impartial moderator. Every person in the group participates openly, and actively, and follows the rules about how such a review is to be conducted. Everyone may get time to express their views, potential faults, and critical areas. After the meeting, a final report is prepared after incorporating necessary suggestions from the moderator.

Advantages of Inspections:

- It can be very effective for finding potential faults or problems in documents like SRS, SDD, etc.
- The critical inspections may also help in finding faults and improve these documents which can in preventing the propagation of a fault in the software development life cycle process.

Disadvantages of Inspections:

- They take time and require discipline.
- It requires more cost and also needs skilled testers.

Applications of Verification Methods

The above three verification methods are very popular and have their strengths and weaknesses. We can compare these methods on various specific issues as given below:

Method	Presenter	No. of Members	Pre-requisites	Report	Strength	Weakness
Peer reviews	0	1 or 2	No prerequisite	Not Required	Less-Expensive	Output is dependent on the ability of the reviewer
Walkthrough	Author	2 to 7 memb	The only presente	The report is pr	Knowledge shari	Find few faults

Method	Presenter	No. of Members	Pre-requisites	Report	Strength	Weakness
		ers	r is required to be prepared	epared by the presenter	ng	and not very expensive
Inspection	Author and other members	3 to 6 members	All participants are required to be prepared	The report is prepared by the moderator	Effective but may get faults	Expensive and requires very skilled membe

Method	Presenter	No. of Members	Pre-requisites	Report	Strength	Weaknesses

Hence, Verification is likely more effective than validation but it may find some faults that are somewhat impossible to find during the validation process. But at the same time, it allows us to find faults at the earliest possible phase/time of software development.

4. **White box testing**
   Link - https://www.geeksforgeeks.org/differences-between-black-box-testing-vs-white-box-testing/

   Basic path testing - https://www.geeksforgeeks.org/basis-path-testing-in-software-testing/

   Graph Matrices - https://www.geeksforgeeks.org/graph-matrices-in-software-testing/

   Loop Testing - https://www.geeksforgeeks.org/loop-software-testing/

5. **Black Box Testing**
   Link - https://www.geeksforgeeks.org/software-engineering-black-box-testing/

   Graph based testing methods
   1.https://testsigma.com/blog/graph-based-testing/
   2.https://www.testbytes.net/blog/black-box-testing/

   Error Guessing - https://www.geeksforgeeks.org/error-guessing-in-software-testing/

   https://www.geeksforgeeks.org/software-testing-boundary-value-analysis/

   https://www.geeksforgeeks.org/equivalence-partitioning-method/

unit 3

https://www.geeksforgeeks.org/difference-between-alpha-and-beta-testing/

https://www.geeksforgeeks.org/system-testing/