TOPS TECHNOLOGY

Module 2

* What is Exploratory Testing?

Exploratory testing is a type of software testing in which the tester is free to Select any possible methodology to test the software.

* What is traceability matrix?

Traceability matrix is a table type document that is used in the development of software application to trace requirements. It can be used for both forward (from requirement to Design to Coding) and backward (from Coding to Requirement) tracing. It is also known as **Requirement Traceability Matrix (RTM) or Cross Reference Matrix (CRM).**

* What is Boundary value testing?

Boundary value is based on testing the boundary values of valid and invalid partitions. The behaviour at the edge of the equivalence partition is more likely to be incorrect than the behaviour within the partition, so boundaries are an area where testing is likely to yield defects.

* What is Equivalence partitioning testing?

Equivalence partitioning is a technique of software testing in which input data is divided into partitions of valid and invalid values, and it is mandatory that all partitions must exhibit the same behaviour. If a condition of one partition is true, then the condition of another equal partition must also be true, and if a condition of one partition is false, then the condition of another equal partition must also be false. The principle of equivalence partitioning is testing cases should be designed to cover each partition at least once. Each value of every equal partition must exhibit the same behaviour as other.

* What is Integration testing?

Integration testing is the second level of the software testing process comes after unit testing. In this testing, units or individual components of the software are tested in a group. The focus of the integration testing level is to expose defects at the time of interaction between integrated components or units.

* What determines the level of risk?
* Example of project risk is Senior Team Member leaving the project abruptly.
* Every risk is assigned a likelihood i.e. chance of it occurring, typically on a scale of 1 to 10. Also, the impact of that risk is identified on a scale of 1-10. But just identifying the risk is not enough. You need to identify mitigation.
* In this case mitigation could be Knowledge Transfer to other team member & having a buffer tester in place example of product risks would be flight reservation system not installing in test environment mitigation in this case would be conducting a smoke or sanity testing.
* Accordingly, you will make changes in your scope items to include sanity testing.

* What is Alpha testing?

Alpha testing is conducted in the organization and tested by a representative group of end-users at the developer's side and sometimes by an independent team of testers.

Alpha testing is simulated or real operational testing at an in-house site. It comes after the unit testing, integration testing, etc. Alpha testing used after all the testing are executed.

It can be a white box, or Black-box testing depends on the requirements - particular lab environment and simulation of the actual environment required for this testing.

* What is beta testing?

Beta testing is the process of testing a software product or service in a real-world environment before its official release. It is an essential step in the software development lifecycle as it helps identify bugs and errors that may have been missed during the development process.

* What is component testing?

A software application is a combination of multiple components or modules. Testing the individual component of any application is called component testing. In this testing, the functionality and usability of each component are validated separately without integrating it with other components, also called module testing.

* What is functional system testing?

Functional testing is a type of testing that seeks to establish whether each application feature works as per the software requirements.

* What is Non-Functional Testing?

Non-Functional Testing is a type of testing used to evaluate a software application’s performance, usability, dependability, and other non-functional characteristics.

* What is GUI Testing?

GUI stands for Graphical User Interface. GUI testing is the process for ensuring proper functionality of the Graphical User Interface for a specific application. GUI testing generally evaluates a design of elements such as layout, colours and also fonts, font size, label, text boxes, text formatting, captions, buttons, list, icons, link and content. GUI testing process may be either manual or automatic and are often performed by third party companies, rather than developers or end users.

* What is Adhoc testing?

Adhoc testing is a type of software testing that is performed informally and randomly after the formal testing is completed to find any loophole in the system. For this reason, it is also known as Random or Monkey testing. Adhoc testing is not performed in a structured way so it is not based on any methodological approach. That’s why Adhoc testing is type of Unstructured Software testing.

* What is load testing?

Load testing where we check an application’s performance by applying some load, which is either less than or equal to the desired load.

Here, load means that when N-number of users using the application simultaneously or sending the request to the server at a time.

* What is stress Testing?

Stress testing is a software testing technique that evaluates the system or application under extreme workloads. This testing helps to identify the system’s breaking points where it fails to responds or crashes. It also helps to identify performance issues like slowdown, memory leaks, and other unexpected behaviours.

* What is white box testing and list the types of white box testing?
  + White box testing, also known as clear box or glass box testing, is a software testing technique that examines the internal structures of a program.
* It involves looking at the code, internal logic, and data flow to ensure the software behave as expected.
* By understanding how the system work inside, testers can create and run more effective test scenarios.

**TYPES:**

1. Statement Coverage – Ensures every line of codes is executed.
2. Branch/ Decision Coverage – Test all possible paths, including if-else conditions.
3. Combination coverage: Check all logical conditions.
4. Path coverage: Ensures all potential paths in the code are tested.

* What is black box testing? What are the different black box testing techniques?
* Black box testing is a software testing method used to test a system without having any prior knowledge of the internal structure of the software under test. Instead, the focus is on how the system works as a whole.
* Specifically, this testing approach focuses on the input that goes into the software the expected results are achieved.

**Black box testing techniques**

1. Equivalence Partitioning (E.P)
2. Boundary Values analysis (BVA)
3. Decision tables
4. State transition testing

* Mention what are the categories of defects?

1. **Data quality/ Database Defects**
2. **Critical Functionality Defects**
3. **Functionality Defects**
4. **Security defects**
5. **User interface Defects**

* Mention what big bang testing is?

Big bang testing is a type of integration testing that combines all the components of a system into one unit and test them as a whole.

* What is the purpose of exit criteria?

Exit criteria in software testing are a set of predefined conditions or requirements that must be met before a software testing phase or software testing project can be considered complete and software can move on to the next stage of development or release.

* When should "Regression Testing" be performed?

1. When brand-new features are introduced
2. When a request for a modification is made (to existing features)
3. When the bag is repaired or the code is refactored
4. Resolve performance issues as soon as possible.
5. When there is a shift in the environment condition.

* What is 7 key principles? Explain in detail?
* **Testing shows presences of defects, not their absences**

We test software to discover issues, so that they can be fixed before they are deployed to live environments – this enables us to have confidence that our systems are working. However, this testing process does not confirm that any software is completely correct and completely devoid of issues. Testing helps greatly reduce the number of undiscovered defects hiding in software, but finding and resolving these issues is not itself proof that the software or system is 100% issue-free.

* **Exhaustive testing is impossible**

it is impossible to test EVERYTHING – ANYTHING – all combinations of inputs and preconditions that attempting to do so is not an efficient use of time and budget.

* **Early testing**

Early testing is often referred to as shift left. Early testing in the software development lifecycle helps to reduce or even completely avoid costly changes. Early testing (and testing in general) helps a company to be more economically successful!

* **Defect clustering**

This principle is an example of the 80:20 rule (also called the Pareto principle) – 80 percent of defects are due to 20 percent of code. While most believe this is some divine mandate, it is based on the observation that 80 percent of users use 20 percent of the software. It is this 20 percent of the software that will contribute most towards the defects.

* **The pesticide paradox**

if the same tests are run continuously then – while they might confirm the software is working – eventually they will fail to find new issues.

* **Testing is Context dependent**

Testing is ALL about the context. The methods and types of testing carried out can completely depend on the context of the software or systems – for example, an e-commerce website can require different types of testing and e-commerce mobile application can requires different types of testing.

* **Absence of error fallacy**

If your software or system is unusable (or does not fulfill users’ wishes) then it does not matter how many defects are found and fixed – it is still unusable. So in this sense, it is irrelevant how issue- or error-free your system is; if the usability is so poor users are unable to navigate, or/and it does not match business requirements then it has failed.

* Difference between QA v/s QC v/s Tester

**QA vs QC vs Tester**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.N.** |  | **Quality Assurance** | **Quality Control** | **Testing (QE)** |
| 1 | Purpose | Prevent issues through establishing quality standards | Verify that the product meets requirement | Detect and fix quality issues |
| 2 | Focus | Development processes | Finished code as a whole | Different aspects of the product: functionality, integrations, performance, etc. |
| 3 | Who | External stakeholders, business analysts, QA engineers, software developers | QA engineers, software developers | QA engineers, software developers |
| 4 | When | Throughout the entire product development life cycle | Before the release | At the testing stage or along the development process |
| 5 | Doing what | Introducing standards, creating guidelines, improving development processes | Validating the product against requirements | Reviewing code, running test, addressing defects |

* Difference between Smoke and Sanity?

|  |  |  |
| --- | --- | --- |
| No. | Smoke Testing | Sanity Testing |
| 1. | Smoke testing is done to assure that the acute functionalities of program is working fine. | Sanity testing is done to check the bugs have been fixed after the build. |
| 2. | Smoke testing is also called subset of acceptance testing. | Sanity testing is also called subset of regression testing. |
| 3. | Smoke testing is documented. | Sanity testing isn’t documented. |
| 4. | Smoke testing is performed by either developers or testers. | Sanity testing is normally performed by tester. |
| 5. | Smoke testing maybe stable or unstable. | Sanity testing is stable. |
| 6. | Smoke testing is scripted. | Sanity testing is usually not scripted. |
| 7. | Smoke testing is done to measure the stability of the system/ product by performing testing. | Sanity testing is done to measure the rationality of the system/ product by performing testing. |
| 8. | Smoke testing is used to test all over function of the system/product. | Sanity testing is used in the case of only modified or defect functions of the system/ products. |
| 9. | Smoke testing can be performed either manually or by using automation tools. | Sanity testing commonly executed manually, not by using any automation approach. |
| 10. | Smoke testing is performed when new product is built. | It includes only those modules where change is code made. |
| 11. | Smoke testing firstly performs on the initial build. Smoke testing is done first. | Sanity testing is done on stable builds or for the introduced new features in the software. |
| 12. | Smoke testing can be carried out either way-manually or automatically | Without using test cases or scripts sanity testing can be carried out. |
| 13. | There is end-to-end system verification done in smoke testing. | A specific component gets verified in sanity testing. |
| 14. | In the smoke testing process, the software build could be stable or unstable | During sanity testing, the software build is comparatively stable. |
| 15. | For every new build release smoke testing is carried out. | Sanity testing is carried out when in-depth testing is not possible because of the short time. |

* Difference between verification and Validation

|  |  |  |  |
| --- | --- | --- | --- |
| S.N. | Criteria | Verification | Validation |
| 1. | Definition | Verification refers to the set of activities that ensure software correctly implements the specific function | Validation refers to the set of activities that ensure that the software that has been built is traceable to customer requirements. |
| 2. | Focus | It includes checking documents, designs, codes, and programs. | It includes testing and validating the actual product. |
| 3. | Type of Testing | Verification is the static testing. | Validation is dynamic testing. |
| 4. | Execution | It does not include the execution of the code. | It includes the execution of the code. |
| 5. | Methods Used | Methods used in verification are reviews, walkthroughs, inspections and desk-checking. | Methods used in validation are Black Box Testing, White Box Testing and non-functional testing. |
| 6. | Purpose | It checks whether the software conforms to specification or not. | It checks whether the software meets the requirements and expectations of a customer or not. |
| 7. | Bug | It can find the bugs in the early stage of the development | It can only find the bugs that could not be found by the verification process. |
| 8. | Goal | The goal of verification is application and software architecture and specification. | The goal of validation is an actual product. |
| 9. | Responsibility | Quality assurance team does verification | Validation is executed on software code with the help of testing team. |
| 10. | Timing | It comes before validation | It comes after verification |
| 11. | Human or computer | It consists of checking of documents/files and is performed by human. | It consists of execution of program and is performed by computer. |
| 12. | Lifecycle | After a valid and complete specification, the verification is starts. | Validation begins as soon as project starts. |
| 13. | Error Focus | Verification is for prevention of errors. | Validation is for detection of errors. |
| 14. | Another Terminology | Verification is also termed as white box testing or static testing as work product goes through reviews. | Validation can be termed as black box testing or dynamic testing as work product is executed. |
| 15. | Performance | Verification finds about 50 to 60% of the defects. | Validation finds about 20 to 30% of the defects. |
| 16. | Stability | Verification is based on the opinion of reviewer and may change from person to person. | Validation is based on the fact and is often stable. |

* Explain types of Performance testing.

1. **Load testing**
2. **Stress testing**
3. **Endurance testing**
4. **Spike testing**
5. **Volume testing**
6. **Scalability testing**

* What is Error, Defect, Bug and failure?
* **Error**: Error is a situation that happens when the Development team or the developer fails to understand a requirement definition and hence that misunderstanding gets translated into buggy code. This situation is referred to as an Error and is mainly a term coined by the developers.
* **Defect**: A defect refers to a situation when the application is not working as per the requirement and the actual and expected result of the application or software are not in sync with each other.
* **Bug:**  A bug refers to defects which means that the software product or the application is not working as per the adhered requirements set. When we have any type of logical error, it causes our code to break, which result in a bug. It is now that the Automation/ Manual Test Engineers describe this situation as bug.
* **Failure:** Failure is the accumulation of several defects that ultimately lead to software failure and result in the loss of information in critical modules thereby making the system unresponsive. Generally, such situation happens very rarely because releasing a product all possible scenarios and test cases for the code are simulated. Failure is detected by end -user once they face a particular issue in the software.
* Difference between Priority and Severity

|  |  |  |
| --- | --- | --- |
| **Features** | **Severity** | **Priority** |
| Definition | Severity is a parameter to denote the impact of a particular defect on the software. | Priority is a parameter to decide the order in which defects should be fixied. |
| Purpose | Severity means how severe the defect is affecting the functionality. | Priority means how fast the defect has to fixied. |
| Relation | Severity is related to the quality standard. | Priority is related to scheduling to resolve the problem. |
| Categories | Severity is divided into 4 categories:   * Critical * Major * Medium * Low | Priorities is divided into 3 categories:   * Low * Medium * High |
| Who decides defects | The testing engineer decides the severity level of the defects. | The product manager decides the priorities of defects. |
| Value | Its value is objective. | Its value is subjective. |
| Value change | Its value doesn’t change from time to time. | Its value changes from time to time. |
| Association | It is associated with functionality of standards. | It is associated with scheduling. |
| Indication | It indicates the seriousness of the bug in the product functionality. | It indicates the how soon the bug should be fixied. |
| Driving factor | It is driven by functionality. | It is driven by business values. |
| Based on | It is based on the technical aspect of the product. | It is based on the customer’s requirements. |

* What is Bug Life Cycle?

**Bug Life Cycle or Defect Life Cycle** in software testing is the specific set of states that defect or bug goes through in its entire life. The purpose of Bug Life Cycle is to easily coordinate and communicate current status of the defect which change to various assignees and make the defect fixing process systematic and efficient.

* Explain the difference between Functional testing and Non-functional testing

|  |  |
| --- | --- |
| **Functional Testing** | **Non-functional Testing** |
| It verifies the operations and actions of an application. | It verifies the behaviour of an application |
| It is based on the requirements of customer. | It is based on expectations of customer. |
| It helps to enhance the behaviour of the applications. | It helps to improve the performance of the application. |
| Functional testing is easy to execute manually. | It is heard to execute non-functional testing manually. |
| It tests what the product does. | It describes how the product does. |
| Functional testing is based on the business requirements. | Non-functional testing is based on the performance requirement. |
| **Examples:**   1. Unit Testing 2. Smoke Testing 3. Integration Testing 4. Regression Testing | **Examples:**   1. Performance Testing 2. Load Testing 3. Stress Testing 4. Scalability Testing |