SOFTWARE TESTING ASSIGNMENT

MODULE – 2 (MANUAL TESTING)

1. What is exploratory testing?

* **Exploratory testing** is a testing where test cases are not created in advance but tester check system on the fly. They may note down ideas about what to test before test execution.
* Exploratory testing is widely used in agile models and is all about discovery, investigation, and learning.

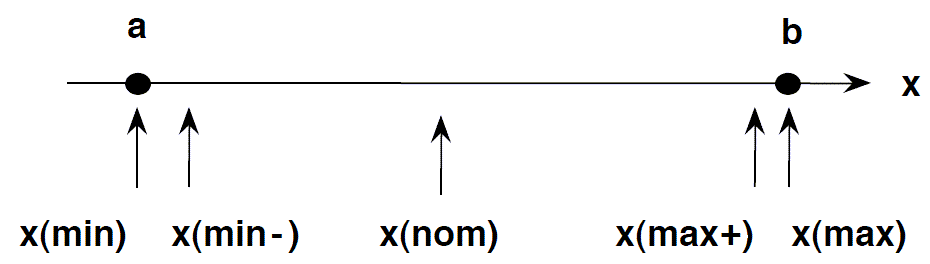
1. What is traceability matrix?

* A **Traceability Matrix** is a document that co-relates any two-baseline documents that require a many-to-many relationship to check the completeness of the relationship.
* It is used to track the requirements and to check the current project requirements are met.
* Types of traceability matrix are –

1. Forward traceability – mapping of requirement to test cases.
2. Backward traceability – mapping of test cases to requirements
3. Bi-directional traceability – A good traceability matrix is references from test cases to basis documentation and vice versa.
4. What is boundary value testing?

* **Boundary value testing** is the process of testing between extreme ends or boundaries between partitions of the input values.
* So these extreme ends like Start- End, Lower- Upper, Maximum-Minimum, Just Inside-Just Outside values are called boundary values and the testing is called “boundary testing”.
* The basic idea in normal boundary value testing is to select input variable values at their:

1. Minimum
2. Just above the minimum
3. A nominal value
4. Just below the maximum
5. Maximum



* In Boundary Testing, Equivalence Class Partitioning plays a good role
* Boundary Testing comes after the Equivalence Class Partitioning.

1. What is equivalence partition testing?

* **Equivalence Partitioning** or Equivalence Class Partitioning is type of black box testing technique which can be applied to all levels of software testing like unit, integration, system, etc. In this technique, input data units are divided into equivalent partitions that can be used to derive test cases which reduce time required for testing because of small number of test cases.
* It divides the input data of software into different equivalence data classes.
* You can apply this technique, where there is a range in the input field.

1. What is integration testing?

* **Integration testing** – testing performed to expose defects in the interfaces and in the interactions between integrated components or systems.
* The purpose of his level of testing is to expose faults in the interaction between integrated units. Test drives and stubs are used to assist in integration testing
* There are 2 levels of integration testing -
  1. component integration testing
  2. system integration testing
* **Component integration testing** is performed to expose defects in the interfaces and interaction between integrated components. It test the interaction between software components and is done after component testing.
* **System integration testing** tests the integration between different systems and may be done after system testing. it

1. What determine the level of risk?

* Level of risk are of two types –

1. Project based risk

* Example of Project risk is Senior Team Member leaving the project abruptly.
* Every risk is assigned 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 members & having a buffer tester in place.

1. Product based risk

* 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.

1. What is alpha testing?

* **Alpha Testing** is a type of acceptance testing; performed to identify all possible issues and bugs before releasing the final product to the end users. Alpha testing is carried out by the testers who are internal employees of the organization.
* The main goal is to identify the tasks that a typical user might perform and test them.
* To put it as simple as possible, this kind of testing is called alpha only because it is done early on, near the end of the development of the software, and before beta testing.
* The main focus of alpha testing is to simulate real users by using a black box and white box techniques.

1. What is beta testing?

* **Beta Testing** is performed by “real users” of the software application in “real environment” and it can be considered as a form of external User Acceptance Testing. It is the final test before shipping a product to the customers.
* Direct feedback from customers is a major advantage of Beta Testing. This testing helps to test products in customer’s environment.
* Beta version of the software is released to a limited number of end-users of the product to obtain feedback on the product quality. Beta testing reduces product failure risks and provides increased quality of the product through customer validation.

1. What is component testing?

* **Component testing** is defined as a software testing type, in which the testing is performed on each individual component separately without integrating with other components.
* Component Testing is also referred to as Unit Testing, Program Testing or Module Testing.
* Generally, any software as a whole is made of several components. Component Level Testing deals with testing these components individually.
* It has two techniques based on depth of testing level –

1. **CTIS – Component Testing In Small**

Component testing may be done with or without isolation of rest of other components in the software or application under test. If it’s performed with the isolation of other component, then it’s referred as Component Testing in Small.

1. **CTIL – Component Testing In Large**

Component testing done without isolation of other components in the software or application under test is referred as Component Testing Large.

1. 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**. Each function is compared to the corresponding requirement to ascertain whether its output is consistent with the end user's expectations.
* The testing is done by providing sample inputs, capturing resulting outputs, and verifying that actual outputs are the same as expected outputs.

1. What is non-functional testing?

* **Non-Functional Testing** is defined as a type of Software testing to check non-functional aspects (performance, usability, reliability, etc) of a software application. It is designed to test the readiness of a system as per nonfunctional parameters which are never addressed by functional testing.
* An excellent example of non-functional test would be to check how many people can simultaneously login into software.
* Non-functional testing is equally important as functional testing and affects client satisfaction.

1. What is GUI testing?

* **GUI Testing** is a software testing type that checks the Graphical User Interface of the Software. The purpose of Graphical User Interface (GUI) Testing is to ensure the functionalities of software application work as per specifications by checking screens and controls like menus, buttons, icons, etc.
* **Approach to GUI testing –**

1. MANUAL BASED TESTING - Under this approach, graphical screens are checked manually by testers in conformance with the requirements stated in business requirements document.
2. RECORD AND REPLAY - GUI testing can be done using automation tools. This is done in 2 parts. During Record, test steps are captured into the automation tool. During playback, the recorded test steps are executed on the Application under Test. Example of such tools - QTP.
3. MODEL BASED TESTING - A model is a graphical description of system’s behavior. It helps us to understand and predict the system behavior. Models help in a generation of eﬃcient test cases using the system requirements.
4. What is adhoc testing?

* **Ad hoc Testing** is an informal or unstructured software testing type that aims to break the testing process in order to find possible defects or errors at an early possible stage. Ad hoc testing is done randomly and it is usually an unplanned activity which does not follow any documentation and test design techniques to create test cases.
* Main aim of this testing is to find defects by random checking. Adhoc testing can be achieved with the Software testing technique called **Error Guessing.**
* Types of adhoc testing –

1. Buddy testing- Two buddies mutually work on identifying defects in the same module. Mostly one buddy will be from development team and another person will be from testing team.

- Buddy testing helps the testers develop better test cases and development team can also make design changes early. This testing usually happens after Unit Testing completion.

1. Pair testing - Pair testing is done only with the testers with different knowledge levels. (Experienced and non-experienced to share their ideas and views)
2. Monkey testing - Randomly test the product or application without test cases with a goal to break the system.
3. What is load testing?

* **Load Testing** is a non-functional software testing process in which the performance of software application is tested under a specific expected load. It determines how the software application behaves while being accessed by multiple users simultaneously.
* The goal of Load Testing is to improve performance bottlenecks and to ensure stability and smooth functioning of software application before deployment.

1. What is stress testing?

* Stress testing - System is stressed beyond its speciﬁcations to check how and when it fails. Performed under heavy load like putting large number beyond storage capacity, complex database queries, continuous input to system or database load.
* Stress testing is used to test the stability & reliability of the system. This test mainly determines the system on its robustness and error handling under extremely heavy load conditions.
* It even tests beyond the normal operating point and evaluates how the system works under those extreme conditions.
* Stress testing is also known as **endurance testing.**

1. **W**hat is white box testing and list the types of white box testing?

* **White Box Testing** is a testing technique in which software’s internal structure, design, and coding are tested to verify input-output flow and improve design, usability, and security. In white box testing, code is visible to testers, so it is also called Clear box testing, open box testing, transparent box testing, Code-based testing, and Glass box testing.
* It is one of two parts of the Box Testing approach to software testing.
* Its counterpart, Black box testing, involves testing from an external or end-user perspective. On the other hand, White box testing in software engineering is based on the inner workings of an application and revolves around internal testing.

- Type of white box testing and its technique –

1) **Web Based Testing:**

- Analyze the logic by reading the code.

- Code optimization Suggest if any optimization can be done in the code which is better than the existing one

- Deploy the code using diﬀerent web servers which could be a case study even if the product is not supporting other web servers.

2) **Desktop Based Testing:**

- When we debug the code when we writing.

3) **Mobile Based Testing:**

- The Android SDK and related plug-in for Eclipse

- Android devices enabled for development and debugging, with appropriate USB drivers as necessary.

4) **Game Based Testing:**

- When we connect with remote device, so which device we connect will check in code.

- When some debug the code and play at that time game.

----------------- Its techniques --------------------------------

1. **Statement Coverage** - The statement coverage is also known as line coverage or segment coverage.

- The statement coverage covers only the true conditions. - Through statement coverage we can identify the statements executed and where the code is not executed because of blockage.

- In this process each and every line of code needs to be checked and executed.

1. **Decision coverage or branch coverage** - Decision coverage also known as branch coverage or all-edges coverage.

- It covers both the true and false conditions unlikely the statement coverage.

- A branch is the outcome of a decision, so branch coverage simply measures which decision outcomes have been tested.

- Aim is to demonstrate that all Decisions have been run at least once.

3) **Condition coverage** - This is closely related to decision coverage but has better sensitivity to the control ﬂow.

- However, full condition coverage does not guarantee full decision coverage.

- Condition coverage reports the true or false outcome of each condition.

- Condition coverage measures the conditions independently of each other.

4) Branch Condition testing - Branch Condition Testing requires that the True and False of each Boolean operand is tested (Boolean Operands in this example: If A > 30 and B >= 5)

5) Branch Condition Combination testing - Branch Condition Combination Coverage would require all combinations of Boolean operands to be evaluated

6) Modified Condition Decision testing - modified Condition Decision Coverage requires test cases to show that each Boolean operand can independently affect the outcome of the decision

7) Data flow testing – Data flow testing aims to execute sub-paths from points where each variable in a component is defined to points where it is referenced.

8) Linear Code Sequence and Jump (LCSAJ) testing - LCSAJ testing requires a model of the source code which identifies control flow jumps (where control flow does not pass to a sequential statement).

17) What is black box testing? What are the different black box testing techniques?

- **Black Box Testing** is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths.

- Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications.

- Black box testing is also known as Behavioral Testing.

-------------DIFFERENT TECHNIQUES OF BLACK BOX------------

1. EQUIVALENCE PARTITION –

* Aim is to treat groups of inputs as equivalent and to select one representative input to test them all
* EP can be used for all Levels of Testing.
* Equivalence partitioning is the process of defining the optimum number of tests by:

1) Reviewing documents such as the Functional Design specification and Detailed Design specification and identifying each input condition within a function,

2) Selecting input data that is representative of all other data that would likely invoke the same process for that particular condition.

1. BOUNDARY VALUE ANALYSIS –

* **Boundary value testing** is the process of testing between extreme ends or boundaries between partitions of the input values.
* So these extreme ends like Start- End, Lower- Upper, Maximum-Minimum, Just Inside-Just Outside values are called boundary values and the testing is called “boundary testing”.

1. DECISION TABLE –

* Inputs are usually defined in terms of actions which are Boolean (true or false).
* Outputs are recorded against each unique combination of inputs.
* Using the Decision Table the relationships between the inputs and the possible outputs are mapped together.
* As with State Transition testing, an excellent tool to capture certain types of system requirements and to document internal system design.
* As such can be used for a number of test levels.
* Especially useful for complex business rules.

1. STATE TRANSITION TESTING –

* State transition testing is used where some aspect of the system can be described in what is called a ‘finite state machine’. This simply means that the system can be in a (finite) number of different states, and the transitions from one state to another are determined by the rules of the ‘machine’. This is the model on which the system and the tests are based.
* Any system where you get a different output for the same input, depending on what has happened before, is a finite state system.
* A finite state system is often shown as a state diagram.

1. USE CASE TESTING –
2. OTHER BLACK BOX TESTING(SYNTAX OR PATTERN TESTING)
3. Mention what are the categories of defects?
4. Mention what big bang testing is?

* In **Big Bang integration** testing all components or modules is integrated simultaneously, after which everything is tested as a whole.
* Big Bang testing has the advantage that everything is finished before integration testing starts.
* The major disadvantage is that in general it is time consuming and difficult to trace the cause of failures because of this late integration.
* Here all component are integrated together at once, and then tested.
* ---------------**Advantages and Disadvantages**--------------------
* Advantages:

1. Convenient for small systems.

* Disadvantages:

1. Fault Localization is difficult.
2. Given the sheer number of interfaces that need to be tested in this approach, some interfaces links to be tested could be missed easily.
3. Since the integration testing can commence only after “all” the modules are designed, testing team will have less time for execution in the testing phase.
4. Since all modules are tested at once, high risk critical modules are not isolated and tested on priority. Peripheral modules which deal with user interfaces are also not isolated and tested on priority.
5. What is the purpose of exit criteria?

* **Exit Criteria** defines the items that must be completed before testing can be concluded
* Successful Testing of Integrated Application.
* Executed Test Cases are documented.
* All High prioritized bugs fixed and closed
* Technical documents to be submitted followed by release Notes.

1. When should "Regression Testing" be performed?

* There is a **need for regression testing** whenever the code is changed, and you need to determine whether the modified code will affect other parts of the software application.
* Moreover, regression testing is needed when a new feature is added to the software application.
* Regression tests may also be performed when a functional or performance defect/issue is fixed.

1. What are 7 key principles? Explain in detail?
2. **Testing shows presence of Defects.**

- Testing can show that defects are present, but cannot prove that there are no defects.

- Testing reduces the probability of undiscovered defects remaining in the software but, even if no defects are found, it is not a proof of correctness.

- We test to find Faults

- As we find more defects, the probability of undiscovered defects remaining in a system reduces.

2. **Exhaustive Testing is Impossible!**

- Testing everything including all combinations of inputs and preconditions is not possible.

- So, instead of doing the exhaustive testing we can use risks and priorities to focus testing efforts.

- For example: In an application in one screen there are 15 input fields, each having 5 possible values, then to test all the valid combinations you would need 30 517 578 125 (515) tests.

- This is very unlikely that the project timescales would allow for this number of tests.

- So, accessing and managing risk is one of the most important activities and reason for testing in any project.

3. **Early Testing**

- Testing activities should start as early as possible in the software or system development life cycle, and should be focused on defined objectives.

- Testing activities should start as early as possible in the development life cycle

- These activities should be focused on defined objectives outlined in the Test Strategy

- Remember from our definition of Testing, that Testing doesn’t start once the code has been written!

4. **Defect Clustering**

- A small number of modules contain most of the defects discovered during pre-release testing, or are responsible for the most operational failures.

- Defects are not evenly spread in a system.

- They are ‘clustered’.

- In other words, most defects found during testing are usually confined to a small number of modules.

- Similarly, most operational failures of a system are usually confined to a small number of modules.

5. **The Pesticide Paradox**

- If the same tests are repeated over and over again, eventually the same set of test cases will no longer find any new defects.

- To overcome this “pesticide paradox”, the test cases need to be regularly reviewed and revised, and new and different tests need to be written to exercise different parts of the software or system to potentially find more defects.

- Testing identiﬁes bugs and programmers respond to fix them.

- As bugs are eliminated by the programmers, the software improves.

- As software improves the effectiveness of previous tests erodes.

6. **Testing is Context Dependent**

- Testing is context dependent which basically means that the way you test an e-commerce site will be different from the way you test a commercial off the shelf application.

- All the developed software’s are not identical.

- You might use a different approach, methodologies, techniques, and types of testing depending upon the application type.

- For instance testing, any POS system at a retail store will be different than testing an ATM machine.

7. **Absence of Errors Fallacy**

- It is possible that software which is 99% bug-free is still unusable. This can be the case if the system is tested thoroughly for the wrong requirement.

- Software testing is not mere finding defects, but also to check that software addresses the business needs.

- The absence of Error is a Fallacy i.e. finding and fixing defects does not help if the system build is unusable and does not fulfill the user’s needs & requirements.

22) Difference between QA v/s QC v/s Tester?

1) **QUALITY ASSURANCE** –

- Activities which ensure the implementation of processes, procedures and standards in context to veriﬁcation of developed software and intended requirements.

- Focuses on processes and procedures rather than conducting actual testing on the system.

- Process oriented activities.

- Preventive activities.

- It is a subset of Software Test Life Cycle (STLC).

2) **QUALITY CONTROL** –

- Activities which ensure the veriﬁcation of developed software with respect to documented (or not in some cases) requirements.

- Focuses on actual testing by executing Software with intend to identify bug/defect through implementation of procedures and process.

- Product oriented activities.

- It is a corrective process.

- QC can be considered as the subset of Quality Assurance.

3) **TESTER** –

- Activities which ensure the identiﬁcation of bugs/error/defects in the Software.

- Focus on actual testing.

- Product oriented activities.

- It is a preventive process.

- Testing is a subset of quality control.

23) Difference between smoke and sanity?

|  |  |  |
| --- | --- | --- |
|  | Smoke Testing | Sanity Testing |
| 1) | Smoke testing is perfomed to ascertain that the critical functionalities of the program is working fine. | Sanity testing is done to check the new functionalities/ bugs have been fixed. |
| 2) | The objective of the testing is to verify the "stability" of the system. | The objective of the testing is to verify the "rationality" of the system in order proceed with more rigorous testing. |
| 3) | This testing is perfomed by devlopers and testers. | This testing is perfomed by testers. |
| 4) | Smoke testing is documented or scripted. | Sanity testing is usually not documented. |

|  |  |  |
| --- | --- | --- |
| 5) | Smoke testig is subset of regression testing. | Sanity testing is subset of acceptance testing. |
| 6) | Smoke testing exercise the entie system from end to end. | Sanity testing exercise in particular component of the entire system. |

24) Difference between verification and Validation?

|  |  |  |
| --- | --- | --- |
| Criteria | Verification | Validation |
| Defination | The process of evaluating work-products (not the actual ﬁnal product) of a development phase to determine whether they meet the speciﬁed requirements for that phase. | The process of evaluating software during or at the end of the development process to determine whether it satisﬁes speciﬁed business requirements. |
| Objective | To ensure that the **product is being built according to the requirements and design speciﬁcations.** In other words, to ensure that work products meet their speciﬁed requirements. | To ensure that **the product actually meets the user’s needs, and that the speciﬁcations** were correct in the ﬁrst place. In other words, to demonstrate that the product fulﬁlls its intended use when placed in its intended environment. |
| Question | Are we building the product right? | Are we building the right product? |
| Evaluation items | Plan , requirement specs,design specs, code ,test case. | The actual product/software. |

-

25) Explain types of Performance testing?

1) load testing - Confirms that the system can handle the required number of users and still operate at a high level of performance.

2) Stress testing - Intentionally tries to break the software by simulating a number of users that greatly exceeds expectations.

3) Endurance testing - The main purpose of endurance testing is to ensure that the application is capable enough to handle extended load without any deterioration of response time.

This type of testing is performed at the last stage of the performance run cycle.

4) Spike testing - Spike testing is a type of performance testing in which an application receives a sudden and extreme increase or decrease in load. The goal of spike testing is to determine the behavior of a software application when it receives extreme variations in traffic.

5) Volume testing - Volume testing comes under software testing. It helps us to check the behavior of an application by inserting a massive volume of the load in terms of data is known as volume testing.

6) Scalability testing - A scalability test is a type of load testing that measures the application's ability to scale up or down as a reaction to an increase in the number of users.

26) What is error, defect, bug and failure?

- We can say that a mistake made by a programmer during coding is called an error.

- An error found during the unit testing in the development phase is called a defect.

- An error found during the testing phase is called a bug.

- When an error is found at an end user's end is called as the failure.

 27) Difference between severity and priority?

|  |  |  |
| --- | --- | --- |
|  | Severity | Priority |
| 1 | Severity is a term that denotes how severely a defect can affect the functionality of the software. | Priority is a term that defines how fast we need to fix a defect. |
| 2 | Severity is basically a parameter that denotes the total impact of a given defect on any software. | Priority is basically a parameter that decides the order in which we should fix the defects. |
| 3 | The testing engineer basically decides a defect’s severity level. | The product manager basically decides a defect’s priority level. |
| 4 | There are 5 types of Severities: Cosmetic, Minor, Moderate, Major, and Critical. | There are 3 types of Priorities: High, Medium, and Low. |

28) What is bug life cycle?



1) New: When a new defect is logged and posted for the first time. It is assigned a status as NEW.

2) Assigned: Once the bug is posted by the tester, the lead of the tester approves the bug and assigns the bug to the developer team

3) Open: The developer starts analyzing and works on the defect fix.

4) Fixed: When a developer makes a necessary code change and verifies the change, he or she can make bug status as “Fixed.”

5) Pending retest: Once the defect is fixed the developer gives a particular code for retesting the code to the tester. Since the software testing remains pending from the testers end, the status assigned is “pending retest.”

6) Retest: Tester does the retesting of the code at this stage to check whether the defect is fixed by the developer or not and changes the status to “Re-test.”

7) Verified: The tester re-tests the bug after it got fixed by the developer. If there is no bug detected in the software, then the bug is fixed and the status assigned is “verified.”

8) Reopen: If the bug persists even after the developer has fixed the bug, the tester changes the status to “reopened”. Once again the bug goes through the life cycle.

9) Closed: If the bug is no longer exists then tester assigns the status “Closed.”

10) Duplicate: If the defect is repeated twice or the defect corresponds to the same concept of the bug, the status is changed to “duplicate.”

11) Rejected: If the developer feels the defect is not a genuine defect then it changes the defect to “rejected.”

12) Deferred: If the present bug is not of a prime priority and if it is expected to get fixed in the next release, then status “Deferred” is assigned to such bugs

13) Not a bug: If it does not affect the functionality of the application then the status assigned to a bug is “Not a bug”.

29) explain the difference between Functional testing and Nonfunctional testing?

|  |  |  |
| --- | --- | --- |
|  | Functional testing | Non functional testing |
| 1) | It is performed before non-functional testing. | It is performed after the functional testing. |
| 2) | It is based on customer’s requirements. | It focuses on customer’s expectation. |
| 3) | It is easy to define functional requirements. | It is difficult to define the requirements for non-functional testing. |
| 4) | Helps to validate the behavior of the application. | Helps to validate the performance of the application. |
| 5) | It describes what the product does. | It describes how the product works. |
| 6) | Check login functionality. | The dashboard should load in 2 seconds. |
| 7) | Examples of Functional Testing Types   * Unit testing * Smoke testing * User Acceptance * Integration Testing * Regression testing * Localization * Globalization * Interoperability | Examples of Non-functional Testing Types   * Performance Testing * Volume Testing * Scalability * Usability Testing * Load Testing * Stress Testing * Compliance Testing * Portability Testing * Disaster Recover Testing |