**Module 2 assignment**

**What is Exploratory Testing?**

Exploratory testing is an approach to software testing that is often described as simultaneous learning, test design, and execution. It focuses on discovery and relies on the guidance of the individual tester to uncover defects that are not easily covered in the scope of other tests

**What is traceability matrix?**

Traceability matrix (TM) is a document, usually in the form of a table, used to assist in determining the completeness of a relationship by correlating any two baselined documents using a many-to-many relationship comparison.

A traceability matrix is a document that details the technical requirements for a given test scenario and its current state. It helps the testing team understand the level of testing that is done for a given product. The traceability process itself is used to review the test cases that were defined for any requirement.

**What is Boundary value testing?**

Boundary value analysis is another black box test design technique and it is used to find the errors at boundaries of input domain rather than finding those errors in the center of input. Equivalence Partitioning and Boundary value analysis are linked to each other and can be used together at all levels of testing.

**What is Equivalence partitioning testing?**

It is a software testing technique that divides the input test data of the application under test into each partition at least once of equivalent data from which test cases can be derived.

An advantage of this approach is it reduces the time required for performing testing of a software due to less number of test cases.

**What is Integration testing?**

Integration testing is known as the second level of the software testing process, following unit testing. Integration testing involves checking individual components or units of a software project to expose defects and problems to verify that they work together as designed.

Software Engineering defines variety of strategies to execute Integration testing, viz.

* Big Bang Approach :
* Incremental Approach: which is further divided into the following
  + Top Down Approach
  + Bottom Up Approach
  + Sandwich Approach – Combination of Top Down and Bottom Up

**What is alpha testing?**

**Alpha Testing** is a type of software testing performed to identify bugs before releasing the software product to the real users or public. It is a type of [acceptance testing.](https://www.guru99.com/user-acceptance-testing.html) The main objective of alpha testing is to refine the software product by finding and fixing the bugs that were not discovered through previous tests.

This testing is referred to as an alpha testing only because it is done early on, near the end of the development of the software, and before Beta Testing.

**Beta testing**

**Beta Testing** is performed by real users of the software application in a real environment. Beta testing is one of the types of **User Acceptance Testing**. A Beta version of the software, whose feedback is needed, is released to a limited number of end-users of the product to obtain feedback on the product quality. Beta testing helps in minimization of product failure risks and it provides increased quality of the product through customer validation. It is the last test before shipping a product to the customers. One of the major advantages of beta testing is direct feedback from customers.

**Characteristics of Beta Testing:**

1. Beta Testing is performed by clients or users who are not employees of the company.
2. Reliability, security, and robustness are checked during beta testing.
3. Beta Testing commonly uses black-box testing.
4. Beta testing is carried out in the user’s location.
5. Beta testing doesn’t require a lab or testing environment.

**What is component testing?**

**Component Testing** is in which usability of each individual component is tested.

It is performed by developer only.

**Objective of Component Testing:**  
The objective of Component Testing is:

* To verify the input and output behavior of the system.
* To check the usability of each component.
* To test the user comprehensibility of the software.
* To test the state of the each components of the system.

Component testing has an important role in finding the issue. Before processing with the integration testing, component testing is performed in order to ensure that each component of the application is working correctly and as per requirement.

**What is functional testing ?**

**FUNCTIONAL TESTING** is a type of software testing that validates the software system against the functional requirements/specifications. The purpose of Functional tests is to test each function of the software application, by providing appropriate input, verifying the output against the Functional requirements.

Types of functional testing

1. black box testing

2. white box testing

3. experienced based testing

4. smoke testing

5. sanity testing

6. end to end testing

7.explotary testing

**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

Types of non functional testing

1. usability testing

2. compactibility testing

3. GUI testing

4. Security testing

5. performance testing

6. load testing

7. stress testing

**What is GUI testing?**

**Graphical User Interface Testing (GUI) Testing** is the process for ensuring proper functionality of the graphical user interface (GUI) for a specific application.GUI testing generally evaluates a design of elements such as layout, colors and also fonts, font sizes, labels, text boxes, text formatting, captions, buttons, lists, icons, links, and content. GUI testing processes may be either manual or automatic and are often performed by third-party companies, rather than developers or end users.

**Feature of Graphical User Interface Testing (GUI):**   
There are some feature of GUI which are given below:

* It provides a customizable test report.
* It is run tests in parallel or distributed on a Selenium Grid with built-in Selenium Webdriver.
* It allows you to test the functionality from a user’s perspective.
* Sometimes the internal functions of the system work correctly but the user interface doesn’t then GUI testing is good to have in addition to the other types.
* It provides reliable object identification, even for web elements with dynamic IDs.

**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

Ad hoc Testing does not follow any structured way of testing and it is randomly done on any part of application. Main aim of this testing is to find defects by random checking.

This testing requires no documentation/ planning /process to be followed. Since this testing aims at finding defects through random approach, without any documentation, defects will not be mapped to test cases. This means that, sometimes, it is very difficult to reproduce the defects as there are no test steps or requirements mapped to it.

**What is load testing?**

A load testing can be done with end-to-end IT systems or smaller components like database servers or firewalls. It measures the speed or capacity of the system or component through transaction response time. When the system components dramatically extend response times or become unstable, the system is likely to have reached its maximum operating capacity. When this happens, the bottlenecks should be identified and solutions provided.

**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 done to make sure that the system would not crash under crunch situations.

• Stress testing is also known as endurance testing.

Need of stress testing

During festival time, an online shopping site may witness a spike in traffic, or when it announces a sale.

• When a blog is mentioned in a leading newspaper, it experiences a sudden surge in traffic.

• To check whether the system works under abnormal conditions.

• Displaying appropriate error message when the system is under stress.

• System failure under extreme conditions could result in enormous revenue loss

• It is better to be prepared for extreme conditions by executing Stress Testing.

**What is white box testing? And list the type of white box testing**

White Box Testing: Testing based on an analysis of the internal structure of the component or system.

White box testing is the detailed investigation of internal logic and structure of the code. • White box testing is also called glass testing or open box testing. In order to perform white box testing on an application, the tester needs to possess knowledge of the internal working of the code. • The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

White box testing involves the testing of the software code for the following:

* Internal security holes
* Broken or poorly structured paths in the coding processes
* The flow of specific inputs through the code
* Expected output
* The functionality of conditional loops
* Testing of each statement, object, and function on an individual basis

**Types of white box testing**

Test/ code coverage

Statement testing

Decision table

**What is black box testing? And different black boxes techniques**

**Black box** testing is a type of software testing in which the functionality of the software is not known. The testing is done without the internal knowledge of the products.

Those tester who do not known the internal structure of an application and by looking on presentation layer they find defects this is called black box testing

**Types of black box testing**

Equivalence partitioning

Boundary value

Decision table state

State transition testing

Use case testing

**What are the categories of defects?**

**Defec**t: Commonly refers to several troubles with the software products, with its external behavior or with its internal features.

**Error of Commission:**  
Commission means instruction or some kind of command given. Now the error in commission means the error in made in command or instruction. For example, suppose I wrote a loop which I was trying to run 10 times but I command it to run more than 10 times by mistake this is the error of commission.

1. **Errors of Omissions:**  
   As name is already describing error of omission is some thing which happens accidentally. Omission word means something left out or executed. Practical most common example of this error is suppose we make a function in programming open its bracket but forget to close at the end.
2. **Error of Clarity:**  
   The most common error in the natural languages. This error happens due to miss understanding between the developer and client. It travels most of the time from the requirements to the software.
3. **Error of Speed or Capacity:**  
   The name of the error is itself enough i think to tell about it this error. Your software is working fine but not working in the required time this is the error of speed. When it comes to capacity it can be relevant to memory. For example, a small integer is declared where the long integer was required.

**What is Big bang testing ?**

Big Bang Testing is an**Integration testing** approach in which all the components or modules are integrated together at once and then tested as a unit. This combined set of components is considered as an entity while testing. If all of the components in the unit are not completed, the integration process will not execute.

Big-bang testing's advantages include the following:

* Its suitability for testing small systems.
* Its ease of identifying errors in such systems, saving time and speeding up application deployment.

However, big-bang testing has disadvantages, for example:

* Locating the source of defects can be difficult since different modules are integrated as one unit.
* Big-bang testing is time-consuming for a large system with numerous units.
* Testers could miss some interface.
* Testers must wait until all modules are available, so they have less time to do the testing and developers have less time to fix any errors.
* Due to simultaneous testing, high-risk critical modules and peripheral modules dealing with user interfaces are not tested on priority (as they should be).

**What is the purpose of exit criteria?**

**Creating exit criteria helps:**

* Align your teams on a common definition of test completion
* Ensure your product meets completion standards before entering the next stage, which avoids costly project delays
* Create clear parameters for test engineers to evaluate software

Purpose of exit criteria

Software testing teams will use exit criteria to determine if a test plan or project can exit to the next stage or be considered complete. This isn't something that should be left up to the subjective and/or ad hoc decisions of a test admin or SQA engineer, as it can directly impact the success of the next stage or project as a whole.

Creating exit criteria helps:

* Align your teams on a common definition of test completion
* Ensure your product meets completion standards before entering the next stage, which avoids costly project delays
* Create clear parameters for test engineers to evaluate software

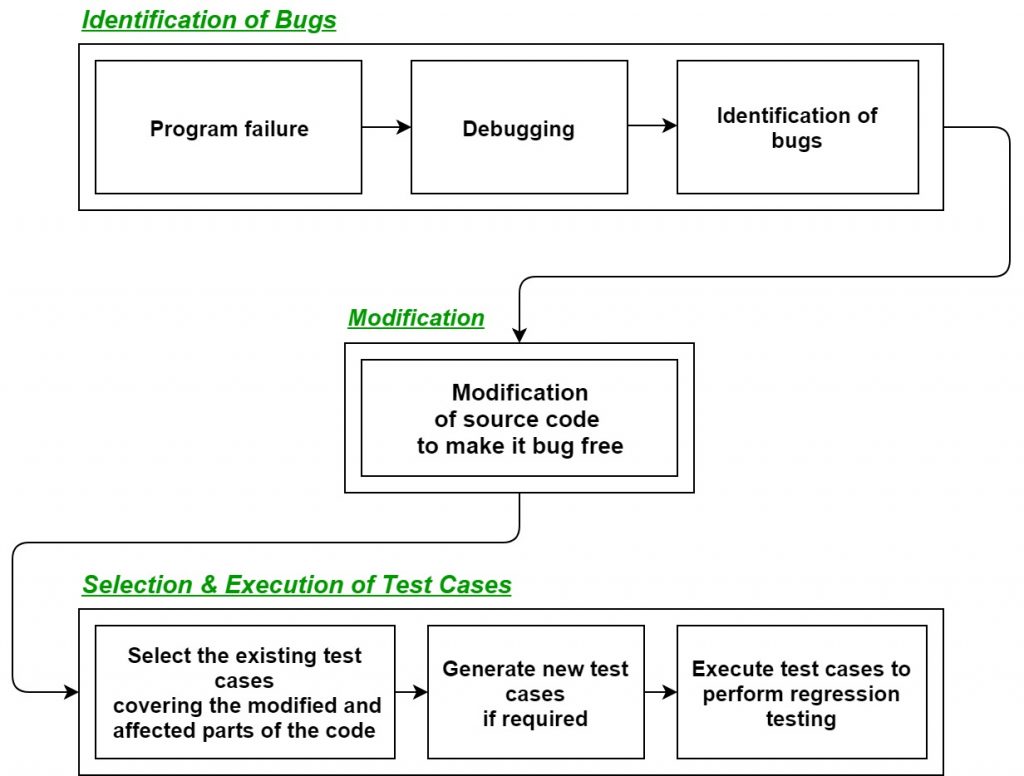
**When do regression testing be performed?**

**Regression Testing**is the process of testing the modified parts of the code and the parts that might get affected due to the modifications to ensure that no new errors have been introduced in the software after the modifications have been made. Regression means return of something and in the software field, it refers to the return of a bug.

**When to do regression testing?**

* When a new functionality is added to the system and the code has been modified to absorb and integrate that functionality with the existing code.
* When some defect has been identified in the software and the code is debugged to fix it.
* When the code is modified to optimize its working.

**Process of Regression testing:**



**What is 7 key principal ? explain in detail.**

### Testing Shows the Presence of Defects, Not Their Absence

Engineers test software to find bugs so they can be repaired before they are released to live environments. By doing this, we can be sure that our systems are operating as intended. However, this testing procedure does not certify that a program is 100% error-free and accurate.

### 2. Exhaustive Testing is Impossible

It is practically impossible to test everything, including all potential combinations of variables and preconditions, despite how much we may want to think or wish it were true. One might also say that trying to do so is not really effective use of time and resources.

Nevertheless, one of the testing abilities is risk assess

### 3. Early Testing Saves Time and Money

Early software testing is crucial to the application lifecycle. This may even entail testing the specifications prior to the start of the coding process; for instance, fixing problems in this phase is much simpler and less expensive than waiting until the very end of the product's lifecycle, when it may be necessary to rewrite entire functionalities, resulting in delays and cost overruns.

Testing is viewed as an ongoing activity instead of a phase (which is a typical waterfall model would be towards the end) since early involvement in testing allows for rapid and efficient ongoing feedback loops. This is another basic Agile concept. Early feedback has been one of the finest methods for a team to overcome obstacles as they arise, and testers are crucial for this. As an important duty to perform, think of the testers as the "information supplier."

### 4. Defects Cluster Together

According to this theory, specific software units or components frequently have the most bugs and cause the majority of operational errors. Due to the predicted and subsequently observed defect prevalence in certain places, testing should thus concentrate on these areas. It is feasible to use the 80:20 Pareto principle, which states that 20% of code is to be blamed for 80% of problems.

### 5. Beware of Pesticide Paradox

This is predicated on the idea that if pesticides are applied to crops frequently enough, insects will ultimately develop an immunity to them, and the pesticide will no longer work. Similarly, if the same procedures are performed repeatedly in testing, they will ultimately stop finding new problems even if they may validate that the product is working.

To help avoid the pesticide paradox, it is crucial to continue to examine the tests and revise or expand existing scenarios. We may accomplish this by running several concurrent testing methodologies, procedures, and approaches concurrently.

### 6. Testing is Context-dependent

Testing is ENTIRELY context-dependent. The procedures and types of testing used might range greatly depending on the environment in which the software or systems are being used. For instance, an e-commerce site may need a different kind of testing than an API service or a database monitoring application. The strategy will always depend on what you are testing.

### 7. Absence-of-errors is a Fallacy

No matter how many flaws are uncovered and patched, if the program or system is impractical (or does not satisfy users' needs), it still isn't useable. Therefore, it makes no difference how problem- or error-free the software is; if the usability is so bad people cannot traverse it, or/and it does not meet business objectives, then it has failed, despite possessing fewer problems. 

**Difference between QA/QC/TESTER**

|  |  |  |  |
| --- | --- | --- | --- |
| **SR NUMBER** | **QA** | **QC** | **TESTER** |
| 1 | Activities which ensure the implementation of processes, procedures and standards in context to verification of developed software and intended requirements. | Activities which ensure the verification of developed software with respect to documented (or not in some cases) requirements. | Activities which ensure the identification of bugs/error/defects in the Software. |
| 2 | Focuses on processes and procedures rather than conducting actual testing on the system. | Focuses on actual testing by executing Software with intend to identify bug/defect through implementation of procedures and process | Focuses on actual testing. |
| 3 | Process oriented activities. | Product oriented activities. | Product oriented activities. |
| 4 | Preventive activities | It is a corrective process. | It is a preventive process. |

**Difference between Smoke and Sanity**

|  |  |
| --- | --- |
| **Smoke testing** | **Sanity testing** |
| The objective of this testing is to verify the “stability” of the system in order to proceed with more rigorous testing | The objective of the testing is to verify the “rationality” of the system in order to proceed with more rigorous testing |
| This testing is performed by the developers or testers | Sanity testing in software testing is usually performed by testers |
| Smoke testing is usually documented or scripted | Sanity testing is usually not documented and is unscripted |
| Smoke testing is a subset of Acceptance testing | Sanity testing is a subset of regression testing. |
| Smoke testing exercises the entire system from end to end | Sanity testing exercises only the particular component of the entire system |
| Smoke testing is like General Health Check Up | Sanity Testing is like specialized health check up |
|  |  |

**Difference between validation and verification**

|  |  |
| --- | --- |
| **verification** | **validation** |
| Verification is the process of checking that a software achieves its goal without any bugs. It is the process to ensure whether the product that is developed is right or not. | **Validation** is the process of checking whether the software product is up to the mark or in other words product has high level requirements |
| It includes checking documents, design, codes and programs. | It includes testing and validating the actual product. |
| Verification is the static testing. | Validation is the dynamic testing. |
| It does *not* include the execution of the code. | It includes the execution of the code. |
| 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. |
| It checks whether the software conforms to specifications or not. | It checks whether the software meets the requirements and expectations of a customer or not. |
| 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. |
| The goal of verification is application and software architecture and specification. | The goal of validation is an actual product. |
| Quality assurance team does verification. | Validation is executed on software code with the help of testing team. |
| It comes before validation. | It comes after verification. |
| It consists of checking of documents/files and is performed by human. | It consists of execution of program and is performed by computer. |
| 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. |
| After a valid and complete specification the verification starts. | Validation begins as soon as project starts. |
| Verification is for prevention of errors. | Validation is for detection of errors. |
| 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. |
| Verification finds about 50 to 60% of the defects. | Validation finds about 20 to 30% of the defects. |
| 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. |
| Verification is about process, standard and guideline. | Validation is about the product. |

**Explain types of performance testing**

**Performance Testing** is a software testing process used for testing the speed, response time, stability, reliability, scalability, and resource usage of a software application under a particular workload. The main purpose of performance testing is to identify and eliminate the performance bottlenecks in the software application.

* **Load testing –** checks the application’s ability to perform under anticipated user loads. The objective is to identify performance bottlenecks before the software application goes live.
* **Stress testing –** involves testing an application under extreme workloads to see how it handles high traffic or data processing. The objective is to identify the breaking point of an application.
* **Endurance testing –** is done to make sure the software can handle the expected load over a long period of time.
* **Spike testing –** tests the software’s reaction to sudden large spikes in the load generated by users.
* **Volume testing** – Under Volume Testing large no. of. Data is populated in a database, and the overall software system’s behavior is monitored. The objective is to check software application’s performance under varying database volumes.
* **Scalability testing**– The objective of scalability testing is to determine the software application’s effectiveness in “scaling up” to support an increase in user load. It helps plan capacity addition to your software system.

**What is error, defects, bugs and failure?**

Errors: The Error is a human mistake. An Error appears not only due to the logical mistake in the code made by the developer.

Defect: A Defect is a variance between expected and actual results. An Error that the tester finds is known as Defect

Failure: Failure is a consequence of a Defect. It is the observable incorrect behavior of the system.

The Bug is the informal name of defects, which means that software or application is not working as per the requirement.

In testing, a software bug can also be issue, error, fault, or failure. The bug occurred when developers made any mistake or error while developing the product.

**Difference between priority and severity?**

|  |  |
| --- | --- |
| **severity** | **priority** |
| 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 fixed. |
| Severity means how severe defect is affecting the functionality. | Priority means how fast defect has to be fixed. |
| Severity is related to the quality standard. | Priority is related to scheduling to resolve the problem. |
| Testing engineer decides the severity level of the defect. | Product manager decides the priorities of defects. |
| Its value is objective. | Its value is subjective. |
| Its value doesn’t change from time to time. | Its value changes from time to time. |
| Severity is of 5 types: Critical, Major, Moderate, Minor, and Cosmetic. | Priority is of 3 types: Low, Medium, and High. |

**What is bug life cycle?**

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

**Difference between functional testing vs non functional testing**

|  |  |
| --- | --- |
| **Functional testing** | **Non functional testing** |
| Functional testing is performed using the functional specification provided by the client and verifies the system against the functional requirement | Non-Functional testing checks the Performance, reliability, scalability and other non-functional aspects of the software system. |
| Functional testing is executed first . | Non functional testing should be performed after functional testing. |
| Manual testing or automation tools can be used for functional testing. | Using tools will be effective for this testing |
| Business requirements are the inputs to functional testing. | Performance parameters like speed , scalability are inputs to non-functional testing. |
| Functional testing describes what the product does | Nonfunctional testing describes how good the product works |
| Easy to do manual testing | Tough to do manual testing |
| Types of Functional testing are  • Unit Testing  • Smoke Testing  • Sanity Testing  • Integration Testing  • White box testing  • Black Box testing  • User Acceptance testing  • Regression Testing | Types of Nonfunctional testing are  • Performance Testing  • Load Testing  • Volume Testing  • Stress Testing  • Security Testing  • Installation Testing  • Penetration Testing  • Compatibility Testing  • Migration Testing |