What is traceability matrix?

* + A traceability matrix is a document that charts the relationship between two baseline documents. There are two of them, one with the test cases and one with the requirement specifications. With its help we are able to trace back from every system component to the original requirement that caused its presence.
  + There are 3 types of Traceability Matrix:

1. Forward Traceability – Mapping of requirements to Test Cases.
2. Backward Traceability – Mapping of Test Cases to Requirements.
3. Bi-directional Traceability – It is the references from the test cases to basis documentation and vice versa.

* What is Boundary value testing?
* Boundary value analysis is a method that improves equivalence partitioning. It is an approach for creating test cases that concentrates software testing effort on examples close to the limits of valid ranges. Compared to equivalence partitioning, boundary value analysis generates test cases that more clearly illustrate mistakes. The secret is to focus software testing efforts towards the extreme ends of the equivalence classes.
* Errors are most likely to arise when input values switch from valid to invalid.In test case design, Boundary Value Approach (BVA) typically works in tandem with EP and employs the same partition analysis.
* Example: Consider a system that accepts numbers from 1 to 99

| Boundary Value Analysis( accepts 1 to 99) | | |
| --- | --- | --- |
| Invalid  (min-1) | Valid  (min, min + 1, nominal, max – 1, max) | Invalid  (max + 1) |
| 0 | 1, 2, 45, 98, 99 | 100 |

* What is Equivalence partitioning testing?
  + It is a software testing technique known as "black-box testing" that categories the input domain into several types of data. From these different types of data, test cases can be produced.
  + The objective is to test all inputs using one representative input and to treat groups of inputs as equivalent.
  + EP is appropriate for all testing levels.
  + Example - If we wish to examine the IF clause "If value is between" i.e., value >=1 and value =100) between 1 to 100 (inclusive).

| If value >=1 AND value <=100 THEN | | |
| --- | --- | --- |
| Invalid | Valid | Invalid |
| 0 | 1, 2, 45, 98, 99,100 | 101 |

* What is Integration testing?
* Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems. Individual software testing components are merged and tested as a group during the integration testing stage of the process.
* This level of testing aims to reveal flaws in the communication between integrated elements. Integration testing is aided by test stubs and test drivers.
* Integration testing examines how components interact with one another, the operating system, the file system, the hardware, and other components of the system. It also examines interfaces across systems.
* There are 2 levels of Integration Testing

1. Component Integration Testing
2. System Integration Testing

* What determines the level of risk?
* Risk can be defined as – “A factor that could result in future negative consequences; usually expressed as impact and likelihood”. When faults discovered through testing are fixed, the software system's quality improves.
* Risks are of two types

1) Project Risks

2) Product Risk

* The level of risk is determined by-

1. The financial impact of the software's release, including support expenses and potential legal action.
2. Software delivery delays.
3. Potential life-threatening situations (safety critical systems).
4. A risk analysis should be used to decide what to test in each component and, more crucially, what not to test. For instance, a risk that is intolerable would require testing, whereas a risk that is acceptable could not require testing.

* What is component testing?
  + Component testing- The testing of individual software components.
  + Component (Unit) can be defined as – “A minimal software item that can be tested in isolation. It means “A unit is the smallest testable part of software.”
  + The goal is to confirm that each software component operates as intended. Prior to performing Integration Testing, Unit Testing is the first stage of testing.
  + Also known as - Unit Testing, Module Testing or Program Testing.
  + Stubs or drivers may be used for isolation testing of the component. Unit testing is aided by the use of frameworks, drivers, stubs, and mock or false objects.
  + The objective of unit testing is to separate out each component of the software and demonstrate that each component is accurate.
* What is functional system testing?
* Functional System Testing: A requirement that specifies a function that a system or system component must perform.
* Requirements can be expressed in text documents or models.
* There are two types of Test Approach

1. Requirement Based Functional Testing
2. Process Based Testing

* What is Non-Functional Testing?
* Non-Functional Testing: Testing the attributes of a component or system that do not relate to functionality, e.g. reliability, eﬃciency, usability, interoperability, maintainability and portability.
* It is a test of "how" the system functions. At all test levels, non-functional testing is possible.
* System response times are checked and fine-tuned using performance testing. Performance testing aims to bring response times down to a manageable level.
* Hence load testing is done to assess how well a system performs under various loads, or amounts of users accessing the system.
* Example –

1. Web Based Testing:

Identify the software processes that directly inﬂuence the overall performanceof the system.

In website number of user/customer will increase, how the website will handle to every customer/user.

1. Desktop Based Testing:

Numerous other such GUI test cases, the desktop application tester must view guarantee that error messages are instructive and helpful for the client memory, and diﬀerent other issues.

1. Mobile Based Testing:

In mobile, automatically will switch oﬀ without any reason. To stop the application which is not in our hand.

1. Game Based Testing: Conﬁrms workability and stability of the software.

* What is white box testing and list the types 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 in-depth analysis of the code's fundamental logic and structure. The tester must be familiar with the inner workings of the code in order to conduct white box testing on an application.
* The tester must examine the source code to determine which component or section of the code is acting improperly.
* Also known as :

1. Glass testing
2. Open box testing.
3. White box testing

There are 3 types of White Box Testing:

1. Statement Coverage- The terms "line coverage" and "segment coverage" are also used to describe the statement coverage.

Only the True circumstances are covered by the statement coverage. We can determine the statements that are executed and the places in the code where execution is halted due to blockage using statement coverage.

Each line of code must be examined and run individually during this procedure. The purpose is to show that every executable statement has been run at least once.

1. Decision Coverage- Also known as branch coverage or all-edges coverage, decision coverage. Both true and false circumstances unlikely the statement coverage are covered.

Branch coverage just counts the number of decision outcomes that have been tried since a branch is the result of a decision.

1. Condition Coverage - Though it is more sensitive to the control flow than decision coverage, this is closely related to it. The true or false outcome of each condition is reported by condition coverage.The conditions are measured separately from one another by condition coverage.

* What is black box testing? What are the different black box testing techniques?
* Black-box testing: Testing, either functional or non-functional, without reference to the internal structure of the component or system.
* The testers are unaware of the internal organization of the system or component. Black-box testing focuses on what the software does rather than how it does it.
* In a typical black box test, the tester interacts with the user interface of the system by providing inputs and evaluating results without being aware of how or where the inputs are processed. Black Box testing is a method for testing an application without having any knowledge of its internal workings.
* Also known as - Specification based testing, Input / Output Driven Testing.
* Example-
  1. Web Based Testing: Takes more time to execute as testers look for game play issues, graphics issues, audio-visual issues, etc.

Validates whether installation goes smoothly, the app works in minimized mode, the app allows social networking options, supports payment gateways, and many more.

* 1. Desktop Based Testing: Resolution change eﬀect on the application
  2. Mobile Based Testing: In mobile, automatically will switch oﬀ without any reason.
  3. Game Based Testing: the game tester must know how to play the game, utilization of the gamepad, know the game ﬂow and the rules.
* Techniques of Black Box Testing

There are four speciﬁcation-based or black-box testing :

* 1. Equivalence partitioning
  2. Boundary value analysis
  3. Decision tables
  4. State transition testing
* 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 benefit that everything is completed prior to the start of integration testing.
* The main drawback of this late integration is that all components are integrated together at once and tested, which generally makes it time consuming and challenging to identify the root cause of errors.
* Advantages :

1. Convenient for small systems.

* Disadvantages :

1. Fault Localization is difficult.
2. Given the enormous number of interfaces that must be evaluated in this approach, some interface links that need to be examined could be easily ignored.
3. The testing team will have less time for execution in the testing phase since integration testing cannot begin until "all" the modules have been designed.
4. Since all modules are tested simultaneously, high risk critical modules are not segregated and tested on priority.

* What is the purpose of exit criteria?
* Exit criteria are used to assess if a particular test activity has been finished or NOT. Exit criteria can be established for each stage of the testing process, including preparation, elaboration, and execution.
* What is 7 key principles? Explain in detail?

The 7 key principles are as follows :

1. Testing shows presence of Defects- Testing cannot establish the absence of flaws, but it can demonstrate their presence.

• Although testing lowers the likelihood that bugs will go undetected and remain in the software, even if no bugs are found, it is not confirmation that the product is accurate.

1. Exhaustive Testing is Impossible.- • It is impossible to test everything, including all potential input and precondition combinations. Therefore, we can use risks and priorities to concentrate testing efforts rather than performing exhaustive testing.
2. Early Testing.- The development lifecycle should begin with testing efforts as soon as possible. The Test Strategy's listed objectives should be the main emphasis of these efforts.
3. Defect Clustering.- A small number of modules either have the majority of the flaws found during pre-release testing or are in charge of the majority of operational errors.Defects in a system are "clustered," not distributed uniformly.
4. The Pesticide Paradox.- If identical tests are conducted repeatedly over territory The same set of test cases will eventually stop uncovering any brand-new flaws. The test cases need to be continually reviewed and improved, and new and different tests need to be produced to exercise various components of the program or system in order to potentially find more flaws, in order to get over the "pesticide paradox."
5. Testing is Context Dependent.- Testing is done diﬀerently in diﬀerent contexts. Diﬀerent kinds of sites are tested diﬀerently. For example -Safety– critical software is tested diﬀerently from an E-commerce site.
6. Absence of Errors Fallacy.- Finding and fixing flaws is useless if the system is unworkable and fails to meet user requirements and expectations. If we create a system and discover and correct flaws while doing so, it doesn't make it a good system.

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

|  |  |  |  |
| --- | --- | --- | --- |
| S.no | Quality Assurance | Quality Control | Testing |
| 1 | Activities which ensure the implementation of procedure which are intended to verify the developed software | Activities which ensure the verification of developed software with respect to documented requirements | Activities which ensure the identification of bugs and errors |
| 2 | Focuses on process and procedures. | Focuses on actual testing through implementation of procedures. | Focuses on actual testing. |
| 3 | Process oriented activities. | Product oriented activities. | Product oriented activities |
| 4 | Preventive Activities. | Corrective process | Preventive process. |
| 5 | Subset of STLC | Subset of QA. | Subset of QC. |

* Difference between verification and Validation.

| Verification | Validation |
| --- | --- |
| It includes checking documents, design, codes and programs. | It includes testing and validating the actual product. |
| 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. |
| 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. |

* What is Error, Defect, Bug and failure?
* Error : A discrepancy between a computed, observed, or measured value or condition and the true, speciﬁed, or theoretically correct value or condition.
* Defect : Commonly refers to several troubles with the software products, with its external behavior or with its internal features.
* Bug : A fault in a program which causes the program to perform in an unintended or unanticipated manner.
* Failure : The inability of a system or component to perform its required functions within speciﬁed performance requirements.
* 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 requirements of customer. | It is based on expectations of customer. |
| It helps to enhance the behaviour of the application. | It helps to improve the performance of the application. |
| Functional testing is easy to execute manually. | It is hard 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 requirement. | 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 |

* + What is the difference between the STLC (Software Testing Life Cycle) and SDLC (Software Development Life Cycle)?

| SDLC | STLC |
| --- | --- |
| SDLC is mainly related to software development. | STLC is mainly related to software testing. |
| Besides development other phases like testing is also included. | It focuses only on testing the software. |
| SDLC involves total six phases or steps. | STLC involves only five phases or steps. |
| In SDLC, more number of members (developers) are required for the whole process. | In STLC, less number of members (testers) are needed. |
| Goal of SDLC is to complete successful development of software. | Goal of STLC is to complete successful testing of software. |
| It helps in developing good quality software. | It helps in making the software defects free. |
| SDLC phases are completed before the STLC phases. | STLC phases are performed after SDLC phases. |
| Creation of reusable software systems is the end result of SDLC. | A tested software system is the end result of STLC. |

* + Explain what Test Plan is? What is the information that should be covered.
* A test plan is a document that lists all upcoming testing-related tasks. It is developed at the project level and, in general, specifies the work items that will be tested, how they will be evaluated.
* Following should be included :
  1. To decide what needs to be tested and what shouldn't be tested in terms of scope and risk.
  2. Test strategy documentation.
  3. Ensuring the testing activities have been covered.
  4. Deciding on entry and exit requirements.
  5. Reviewing the test result.
  6. Organizing the timing and method of the testing, choosing how the test findings will be assessed, and determining the test exit criteria.
  7. Ensuring that the test documentation produces reliable test assets.
* Difference between smoke and sanity

|  |  |
| --- | --- |
| Smoke Testing is performed to ascertain that the **critical functionalities** of the program are **working fine.** | Sanity testing is done at random to verify that **each functionality** is **working as expected.** |
| Smoke testing exercises the **entire system** from end to end. | Sanity testing exercises only the **particular component** of the entire system. |
| The main objective of the testing is to verify the **stability** of the system. | The main objective of the testing is to verify the **rationality** of the system. |
| Smoke testing is usually **documented and scripted.** | Sanity testing **is not** documented and is unscripted. |
| This testing is performed by the **developers or testers.** | Sanity testing in software testing is usually performed by **testers.** |
| It is a well **elaborate and planned** testing. | This **is not a planned** test and is done only when there is a shortage of time. |
| This is a **wide** and **deep** testing. | This is a **wide** and **shallow** testing. |
| Smoke testing is a subset of [Acceptance testing.](https://www.practitest.com/qa-learningcenter/best-practices/what-is-uat-testing/) | Sanity testing is a subset of **Regression Testing.** |

s