**Module 2**

1. **What is software testing?**

* **Software Testing** is a method to check whether the actual software product matches expected requirements and to ensure that software product is[Defect](https://www.guru99.com/defect-management-process.html)free. It involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements

1. **What is 7 key principle? explain in details.**

* Here are 7 key principle of testing.

1. Testing shows presence of defects
2. Exhaustive testing is not possible
3. Early testing
4. Defect clustering
5. Pesticide paradox
6. Testing is context dependent
7. Absence of errors fallacy
8. **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.

1. **Exhaustive testing is not possible:**

* 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.
* We have learned that we cannot test everything (i.e. all combinations of inputs and preconditions).
* That is we must Prioritise our testing effort using a Risk Based Approach.

1. **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!

1. **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 numbers of module.

1. **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
* Therefore we must learn, create and use new tests based on new techniques to catch new bugs.

1. **Testing is Context Dependent:**

* Testing is basically 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.
* Whilst, Testing can be 50% of development costs, in NASA's Apollo program it was 80% testing
* 3 to 10 failures per thousand lines of code (KLOC) typical for commercial software • 1 to 3 failures per KLOC typical for industrial software
* 0.01 failures per KLOC for NASA Shuttle code!
* Also different industries impose different testing standards

1. **Absence of Errors Fallacy:**

* If the system built is unusable and does not fulfill the user’s needs and expectations then finding and fixing defects does not help.
* If we build a system and, in doing so, find and fix defects....
* It doesn’t make it a good system
* Even after defects have been resolved it may still be unusable and/or does not fulfil the users.

1. **What is error, defect, bug and failure?**

* Basically “A mistake in coding is called error, error found by tester is called defect, defect accepted by development team then it is called bug, build does not meet the requirements then it is failure”.in advance,
* **Error:** A Human can make an Error . An Error is ‘A Human Action that produces an Incorrect Result’ . The Error can cause a Defect.
* **Defect:** A Defect is ‘A flaw in a component or system that can cause the component or system to fail to perform its required function’ .A Defect can be in the Software, System or in a Document.

Defects occur because human beings are fallible Also because of: time pressure,complex code, complex infrastructure, changed technologies, and/or many system interactions. A Defect may result in a Failure.

* **Bug:** A fault in a program which causes the program to perform in an unintended or unanticipated manner. See: anomaly, defect, error, exception, and fault. Bug is terminology of tester.
* **Failure:** A Failure is a ‘Deviation of the component or system from its expected delivery, service or result’. Failures can be caused by environmental conditions as well

E.g. radiation, magnetism, electronic fields

• Pollution can cause faults in firmware or influence the execution of software by changing hardware condition.

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. no** | **Quality Assurance** | **Quality control** | **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** | It is a preventive process. | It is a corrective process. | It is a preventive process. |
| **5** | It is a subset of Software Test Life Cycle (STLC). | QC can be considered as the subset of Quality Assurance | Testing is the subset of Quality Control. |

1. **What determine the level of risk?**

* A properly designed test that passes, reduces the overall level of Risk in a system
* Risk – ‘A factor that could result in future negative consequences; usually expressed as impact and likelihood’
* When testing does find defects, the Quality of the software system increases when those defects are fixed.
* A Risk could be any future event with a negative consequence .You need to identify the risks associated with your project.
* Risks are of two types : 1. Project Risks

2. Product Risk

1. **What is the difference between test scenario, test case and test script?**
2. **Test Scenario:**

A Scenario is any functionality that can be tested. It is also called Test Condition, or Test Possibility.

• Test Scenario is ‘What to be tested’

• Test scenario is nothing but test procedure.

• The scenarios are derived from use cases.

• Test Scenario represents a series of actions that are associated together.

1. **Test case:**

Test cases involve the set of steps, conditions and inputs which can be used while performing the testing tasks.

• Test Case is ‘How to be tested’

• Test case consist of set of input values, execution precondition, expected Results and executed post-condition developed to cover certain test Condition.

• Test cases are derived (or written) from test scenario.

• Test Case represents a single (low level) action by the user.

• Test cases are set of input and output given to the System.

The main components which are always available and included in every test case:

Test case ID

• Product Module ID

• Product version (Optional)

• Revision history (Optional)

* Purpose/ Test Case Description
* Assumptions (Optional)
* Pre-Conditions(Optional)
* Test Steps
* Expected Outcome/Result
* Actual Outcome/Result
* Post Conditions(Pass/Fail)
* The Step # Identifies the task sequence in the script

**3 .test script:**

* A set of sequential instruction that detail how to execute a core business function
* One script is written to explain how to simulate each business scenario
* Written to a level of detail for which someone else (other than the script writer ) would be able to easily execute
* Identifies the test condition that is being satisfied for each step, if applicable
* Identified the input/test data that should be entered for each transaction
* Identifies the expected results for each step, if applicable
* Should demonstrate how the system can support the HCA warehouse business processes
* A test script in software testing is a set of instructions that will be performed on the system under test to test that the system functions as expected.
* There are various means for executing test scripts: 1. Manual testing

2.automation testing

1. **Explain what test plan is? What is the information that should be covered?**

* A document describing the scope, approach, resources and schedule of intended test activities. It is a high level document in which how to perform testing is described.

1. Analyze theproduct
2. Design the test strategy
3. Define scope of testing : in scope and out of scope.
4. Identify the testing type
5. Document risk and issues
6. Create test logistic
7. Define the test objective
8. Define the test criteria: suspension and exit
9. Resource planning : human and system
10. Plan test environment
11. Schedule and estimation
12. Determine test deliverables :deliverables are provide before, during and after testing phase
13. **Difference between verification and validation.**

|  |  |  |
| --- | --- | --- |
| **criteria** | **verification** | **validation** |
| **Definition** | 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** | Plans, Requirement Specs, Design Specs, Code, Test Cases | The actual product/software |
| **Activities** | • Reviews  • Walkthroughs  • Inspections | • Testing |

1. **What is the difference between SDLC and STLC.**

|  |  |  |
| --- | --- | --- |
| **Parameter** | **SDLC** | **STLC** |
| **Origin** | Software development life cycle. | Software testing life cycle |
| **Objective** | The main object of SDLC life cycle is to complete successful development of the software including testing and other phases | The only objective of the STLC phase is testing. |
| **Requirement Gathering** | In SDLC the business analyst gathers the requirements and create Development Plan | In STLC, the QA team analyze requirement documents like functional and non-functional documents and create System Test Plan |
| **High & Low-Level Design** | the development team creates the high and low-level design plans | the test analyst creates the Integration Test Plan |
| **Coding** | The real code is developed, and actual work takes place as per the design documents. | The testing team prepares the test environment and executes them |
| **Maintenance** | SDLC phase also includes post-deployment supports and updates. | Testers, execute regression suits, usually automation scripts to check maintenance code deployed. |

1. **What is integration testing?**

* Integration Testing - Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems
* Integration Testing is a level of the software testing process where individual units are combined and tested as a group.
* There are 2 levels of Integration Testing :

1. Component Integration Testing

2. System Integration Testing

There is two types methods of Integration Testing:

• Bing Bang Integration Testing

• Incremental Integration Testing: 1. Top Down Approach

2. Bottom Up Approach

**11. What is big bang testing?**

* In Big Bang integration testing all components or modules is integrated simultaneously, after which everything is tested as a whole.
* **Advantages**: 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

**12. What is alpha testing?**

* It is always performed by the developers at the software development site.
* Sometimes it is also performed by Independent Testing Team.
* Alpha Testing is not open to the market and public
* It is conducted for the software application and project.
* It is always performed in Virtual Environment.
* It is always performed within the organization.
* It is the form of Acceptance Testing
* Alpha Testing is definitely performed and carried out at the developing organizations location with the involvement of developers.
* It comes under the category of both White Box Testing and Black Box Testing. During this phase, the following will be tested in the application: (1) Spelling Mistakes .(2) Broken Links (3)Cloudy Directions
* Alpha Testing is always performed at the time of Acceptance Testing when developers test the product and project to check whether it meets the user requirements or not.
* It is always performed at the developer’s premises in the absence of the users.
* It is considered as the User Acceptance Testing (UAT) which is done at developer’s area.
* Unit testing, integration testing and system testing when combined are known as alpha testing.

**13. What is beta testing?**

* It is always performed by the customers at their own site.
* It is not performed by Independent Testing Team.
* Beta Testing is always open to the market and public.
* It is usually conducted for software product.
* It is performed in Real Time Environment.
* It is always performed outside the organization.
* It is also the form of Acceptance Testing.
* Beta Testing (field testing) is performed and carried out by users or you can say people at their own locations and site using customer data.
* It is only a kind of Black Box Testing.
* Beta Testing is always performed at the time when software product and project are marketed.
* It is always performed at the user’s premises in the absence of the development team.
* It is also considered as the User Acceptance Testing (UAT) which is done at customers or users area.
* Beta testing can be considered “pre-release” testing.

**14. What is component testing?**

* Component (Unit) – A minimal software item that can be tested in isolation. It means “A unit is the smallest testable part of software.”
* Component Testing – The testing of individual software components.
* Unit Testing is a level of the software testing process where individual units/components of a software/system are tested. The purpose is to validate that each unit of the software performs as designed.
* Unit testing is the first level of testing and is performed prior to Integration Testing.
* Sometimes known as Unit Testing, Module Testing or Program Testing
* Component can be tested in isolation – stubs/drivers may be employed
* Unit testing frameworks, drivers, stubs and mock or fake objects are used to assist in unit testing.
* Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended with debugging tool.

**15. What is functional system testing?**

* Functional System Testing : A requirement that specifies a function that a system or system component must perform
* A Requirement may exist as a text document and/or a model
* There is two types of techniques:

1]Requirement Based Functional Testing.

2] Process Based Testing

* **Functional System Testing Functionality As below:**

|  |  |
| --- | --- |
| **Accuracy** | Provision of right or agreed results or effects |
| **Interoperability** | Ability to interact with specified systems |
| **Compliance** | Adhere to applicable standards, conventions, regulations or laws |
| **Auditability** | Ability to provide adequate and accurate audit data |
| **Suitability** | Presence and appropriateness of functions for specified tasks |

1. **Requirement Based Testing**

• Testing against requirements and specifications

• Test procedures and cases derived from:

[1]detailed user requirements

[2] system requirements functional specification

[3]User documentation/instructions

[4] high level System design

• Starts by using the most appropriate black-box testing techniques

• May support this with white-box techniques (e.g. menu structures, web page navigation)

1. **Business Process Based Testing:**

* Testing should reflect the business environment and processes in which the system will operate.
* Therefore, test cases should be based on real business processes.
* Test procedures and cases derived from:

[1]Expected user profiles

[2] Business scenarios

[3]Use cases

**16. Difference between functional testing and Non functional testing.**

|  |  |
| --- | --- |
| **Functional** | **Non functional** |
| 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 this 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 work |
| Types of Functional testing are :  • Unit Testing • Smoke Testing • Sanity Testing • Integration Testing • White box testing • Black Box testing • User Acceptance testing | Types of Nonfunctional testing are :  • Performance Testing • Load Testing • Volume Testing • Stress Testing • Security Testing • Installation Testing • Penetration Testing • Compatibility Testing |

**17. What is Non functional testing?**

* The non-functional aspects of a system are all the attributes other than business functionality, and are as important as the functional aspects. These include:

[1] the look and feel and ease of use of the system

[2] how quickly the system performs

[3] how much the system can do for the user

[4] how easy and quick the system is to install

[5] how robust it is

[6] how quickly the system can recover from a crash

* Testing of those requirements that do not relate to functionality
* Emphasis on non-functional requirements:

[1]Performance [2] Load [3] Data volumes [4] Storage [5] Recovery [6] Usability [7] Stress [8] Security

**18. What is boundary value testing?**

* Boundary value analysis is a methodology for designing test cases that concentrates software testing effort on cases near the limits of valid ranges
* Boundary value analysis is a method which refines equivalence partitioning.
* Boundary value analysis generates test cases that highlight errors better than equivalence partitioning.
* The trick is to concentrate software testing efforts at the extreme ends of the equivalence classes.
* At those points when input values change from valid to invalid errors are most likely to occur.
* Boundary Value Analysis (BVA) uses the same analysis of partitions as EP and is usually used in conjunction with EP in test case design.
* BVA operates on the basis that experience shows us that errors are most likely to exist at the boundaries between partitions and in doing so incorporates a degree of negative testing into the test design.
* BVA Test cases are designed to exercise the software on and at either side of boundary values.

**19. What is Equivalence partitioning testing?**

* 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.
* If we want to test the following IF statement: “If value is between 1 and 100 (inclusive) (e.g value >=1 and value <=100) Then...”
* In EP we must identify Valid Equivalence partitions and Invalid Equivalence partitions where applicable (typically in range tests).
* The Valid partition is bounded by the values 1 and 100
* Plus there are 2 Invalid partitions.
* EP can help reduce the number of tests from a list of all possible inputs to a minimum set that would still test each partition
* If the tester chooses the right partitions, the testing will be accurate and efficient
* EP is used to achieve good input and output coverage, knowing exhaustive testing is often impossible
* It can be applied to human input, input via interfaces to a system, or interface parameters in integration testing.

**20. 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.
* Specification-based testing technique is also known as ‘black-box’ or input/output driven testing techniques because they view the software as a black-box with inputs and outputs.
* The testers have no knowledge of how the system or component is structured inside the box. In black-box testing the tester is concentrating on what the software does, not how it does it.
* Typically, when performing a black box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.
* **Advantages:**

1. Well suited and efficient for large code segments.
2. Code Access not required.
3. Clearly separates user's perspective from the developer's perspective through visibly defined roles.
4. Large numbers of moderately skilled testers can test the application with no knowledge of implementation, programming language or operating systems.

* **Disadvantages:**
  1. Limited Coverage since only a selected number of test scenarios are actuall performed.
  2. Inefficient testing, due to the fact that the tester only has limited knowledge about an application.
  3. Blind Coverage, since the tester cannot target specific code segments or error prone area.

**Techniques of Black Box Testing:**

1. Equivalence partitioning
2. Boundary value analysis
3. Decision tables
4. State transition testing
5. Use-case Testing

**21.**