Assignment -2

Model-2

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

* White box testing based on an analysis of the internal structure of the component or system.
* The testers have no knowledge of how the system or component is structured inside the box.
* Specification-based techniques are appropriate at all levels of testing component testing through to acceptance testing where a specification exits.
* For example, when performing system or acceptance testing, the requirements specification or functional specification many from the basis of the tests.
* Typically used at System Test phase, although can be useful throughout the test lifecycle.
* The tester is oblivious to the system architecture and dose not have access to the source code.

**Types of white box testing**

Top Down testing

Hybrid Approach

Bottom up approach

Unit testing

Integration testing

Execution testing

Operations testing

Mutation testing

Statement Coverage

Brach Coverage

Path Coverage

White Box testing

1. **Unit testing:** - **Unit Testing** is a software testing technique in which individual units or components of a software application are tested in isolation.

* **Unit tests** for these code units and run them automatically every time you make changes.
* Unit testing promotes modular code, ensures better test coverage, and saves time by allowing developers to focus more on coding than manual testing.
* Understanding of the software development process.
* Knowledge of programming languages and development tools.

1. **Integration testing: -** Integration testing is the process of testing the interface between two software units or modules.

* The purpose of integration testing is to expose faults in the interaction between integrated units.
* Integration testing is typically performed after unit testing and before system testing.
* Integration testing is typically performed after unit testing and before system testing.

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

* **Error: -** A human action that produces an incorrect result.
* An error is a mistake, an incorrect action, or a deviation from what is correct.
* Error can occur in many situations, including in speech, writing, calculations.
* **Defect: -** A flaw in a component or system that can cause the component or system to fail to perform its required function.
* A bug is a type of error in software code, while a defect is a broader term that includes bugs and other issues.
* Defect testing is the process of finding and fixing coding errors in software.
* **Failure: -** Deviation of the component or system from its expected delivery, service or result.
* Failure is the social concept of not meeting a desirable or intended objective, and is usually viewed as the opposite of success.
* Failures occur when defects remain undetected or unaddressed and reach the end users.
* **Bug: -** A fault in a program which causes the program which causes the program to perform in an unintended or unanticipated manner.
* In computer technology, a bug is a coding error in a computer program.

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

|  |  |  |  |
| --- | --- | --- | --- |
| S.N. | Quality Assurance | Quality Control | Testing |
| 1 | Activities which ensure the implementation of procedures, 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 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. |
| 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 is 7 key principles? Explain in detail?**

* Software Testing is a crucial part of the Software Development Life Cycle (SDLC), ensuring that the final product meets quality standards. The **7 key principles of software testing** help testers perform efficient and effective testing.

1. **Testing shows presence of defects**
2. **Exhaustive testing is impossible!**
3. **Early testing**
4. **Defect clustering**
5. **The pesticide paradox**
6. **Testing is context dependent**
7. **Absence of Errors fallacy**

1. **Testing shows presence of defects**

* Testing helps in identifying defects in the software but **cannot guarantee** that the software is completely defect-free.
* Even if no defects are found, it does not mean the system is perfect; there may still be hidden issues.
* The goal of testing is to **reduce the number of defects**, not to prove that the software is completely flawless.
* However, Testing cannot prove that there are no defects present.

1. **Exhaustive testing in impossible!**

* It is impossible to test all possible inputs, conditions, and scenarios in a real-world application.
* Instead of testing everything, **risk-based and prioritized testing** is used.
* Testers focus on the most **important functionalities, high-risk areas, and critical features**.
* 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.

1. **Early testing**

* The earlier a defect is found in the SDLC, the cheaper it is to fix.
* Testing should start at the **requirement analysis and design phase** rather than waiting until development is complete.
* Early testing, such as **unit testing and integration testing**, helps in identifying issues before they become costly.
* Testing activities should start as early as possible in the development life cycle.

1. **Defect clustering**

* A small number of modules in a software system often contain the most defects.
* This follows the **Pareto Principle (80-20 rule)**: 80% of the defects come from 20% of the modules.
* Testers focus more on these **high-risk areas** to increase testing efficiency.
* Similarly, most operational failures of a system are usually confined to a small number of modules
* An important consideration in test prioritisation!

1. **The pesticide paradox**

* Running the **same test cases repeatedly** will eventually stop finding new defects.
* To overcome this, test cases should be **regularly reviewed, updated, and modified**.
* New test scenarios should be created to improve defect detection.
* As bugs are eliminated by the programmers, the software improves.

1. **Testing is Context Dependent**

* The approach to testing depends on the **type of software** being tested.
* Testing is done differently in different contexts.
* Different kinds of sites are tested differently.
* For example:
* A banking application requires **high security and precision** testing.
* A gaming application requires **performance and usability** testing.
* The testing strategy must align with the business goals and project requirements.

1. **Absence-of-Errors is a fallacy**

* Even if the software is **100% bug-free,** it may still fail to meet user expectations.
* The software must be **tested against requirements and user needs**, not just for technical correctness.
* A product that is error-free but does not solve the user’s problem is considered a failure.
* Even after defects have been resolved it may still be unusable and/or does not fulfil the users’ needs and expectations.