Name: A. Shiva Surya Saran
Roll No.: BE-B 20

Software Testing and Quality Assurance

Assignment 2

Unit 2: Test Planning and Management

1. Explain term Mutation Testing?

Mutation testing is a structural testing technique, which uses the structure of the code to guide the testing process. On a very high level, it is the process of rewriting the source code in small ways to remove the redundancies in the source code

These ambiguities might cause failures in the software if not fixed and can easily pass through testing phase undetected.

Mutation Testing Benefits:

Following benefits are experienced if mutation testing is adopted:

- It brings a whole new kind of errors to the developer's attention.
- It is the most powerful method to detect hidden defects, which might be impossible to identify using the conventional testing techniques.
- Tools such as Insure++ help us to find defects in the code using the state-of-the-art.
- Increased customer satisfaction index as the product would be less buggy.
- Debugging and Maintaining the product would be easier than ever.

Mutation Testing Types:

- **Value Mutations:** An attempt to change the values to detect errors in the programs. We usually change one value to a much larger value or one value to a much smaller value. The most common strategy is to change the constants.
- **Decision Mutations:** The decisions/conditions are changed to check for the design errors. Typically, one changes the arithmetic operators to locate the defects and also we can consider mutating all relational operators and logical operators (AND, OR , NOT)
- **Statement Mutations:** Changes done to the statements by deleting or duplicating the line which might arise when a developer is copy pasting the code from somewhere else.

2. Explain Software testing principles

There are seven principles in software 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

- **Testing shows presence of defects:** The goal of software testing is to make the software fail. Software testing reduces the presence of defects. Software testing talks about the presence of defects and does not talk about the absence of defects. Software testing can ensure that defects are present, but it cannot prove that software is defects free. Even multiple testing can never ensure that software is 100% bug-free. Testing can reduce the number of defects but not removes all defects.

- **Exhaustive testing is not possible:** It is the process of testing the functionality of a software in all possible inputs (valid or invalid) and pre-conditions is known as exhaustive testing. Exhaustive testing is impossible means the software can never test at every test cases. It can test only some test cases and assume that software is correct, and it will produce the correct output in every test cases. If the software will test every test case then it will take more cost, effort, etc. and which is impractical.

- **Early Testing:** To find the defect in the software, early test activity shall be started. The defect detected in early phases of SDLC will very less expensive. For better performance of software, software testing will start at initial phase i.e. testing will perform at the requirement analysis phase.

- **Defect clustering:** In a project, a small number of the module can contain most of the defects. Pareto Principle to software testing state that 80% of software defect comes from 20% of modules.

- **Pesticide paradox:** Repeating the same test cases again and again will not find new bugs. So, it is necessary to review the test cases and add or update test cases to find new bugs.

- **Testing is context dependent:** Testing approach depends on context of software developed. Different types of software need to perform different types of testing. For example, the testing of the e-commerce site is different from the testing of the Android application.

- **Absence of errors fallacy:** If a built software is 99% bug-free but it does not follow the user requirement then it is unusable. It is not only necessary that software is 99% bug-free but it also mandatory to fulfil all the customer requirements.

3. Design Test Strategy for any suitable example

Test Strategy in STLC:

Scope
It defines parameters like

- Who will review the document?
- Who will approve this document?
- Software Testing activities carried out with timelines

Test Approach

It defines

- Process of testing
- Testing levels
- Roles and responsibilities of each team member
- Types of Testing (Load testing, Security testing, Performance testing etc.)
- Testing approach & automation tool if applicable
- Adding new defects, re-testing, Defect triage, Regression Testing and test sign off

Test Environment

- Define the number of requirement and setup required for each environment
- Define backup of test data and restore strategy

Testing Tools

- Automation and Test management tools needed for test execution
- Figure out a number of open-source as well as commercial tools required, and determine how many users are supported on it and plan accordingly

Release Control

- Release management plan with appropriate version history that will make sure test execution for all modification in that release

Risk Analysis

- List all risks that you can estimate
- Give a clear plan to mitigate the risks also a contingency plan

Review and Approvals

- All these activities are reviewed and signed off by the business team, project management, development team, etc.
- Summary of review changes should be traced at the beginning of the document along with an approved date, name, and comment

4. How to find a test team efficiency?

Test Efficiency can be defined as the most efficient way in which the resources associated with the software project are utilized to achieve an organization goal (for example, number of projects to be completed in that particular year).

Test efficiency is an internal process for an organization that evaluates the utilization of resources which could be a timeline, hardware, manpower, expertise of test team, etc. to develop a software product.

Mathematically test efficiency is calculated as a percentage of the number of alpha testing (in-house or on-site) defects divided by sum of a number of alpha testing and a number of beta testing (off-site) defects. Alpha testing is the testing done by the expertise project test team on-site and this test team is expected to test the product thoroughly before the product is available to the customer or end user in the market.

Once internal alpha testing is completed the product is made available to end users to test and look for the defects and provide their valuable feedback. Ideally, there should not be any defect observed by the end user during beta testing as any valid bug identified during beta testing, it will directly deteriorate the test efficiency of the on-site project team.

Test Efficiency Formula:

Test Efficiency = (Test Defects (Alpha) / (Test Defects (Alpha) + Acceptance Defects (Beta))) * 100

5. Describe important features of testing Process

- Testing is destructive process but it's constructive destruction.
- Testing needs positive approach with a consideration that there is defect present in the code.
- If test doesn't detect any defect present in system, then its unsuccessful testing.
- Root cause analysis and corrective/preventive actions must be mentioned.
- Performing regression testing when defects are resolved by development team
- Proper testing helps to improve overall quality and efficiency of the product.

6. Short note on Test Planning & Test Strategy.

Test Strategy

It is a plan for defining the approach to the Software Testing Life Cycle (STLC). It guides the QA team to define Test coverage and testing scope. It also aids testers to get a clear picture of the project at any instance. The possibility of missing any test activity is very low when there is a proper test strategy in place.

Test Planning

A test plan outlines the strategy that will be used to test an application, the resources that will be used, the test environment in which testing will be performed, and the limitations of the testing and the schedule of the testing activities. Typically, the Quality Assurance Team Lead will be responsible for writing a Test Plan.

A Test Plan includes the following.

- Introduction to the Test Plan document.
- Assumptions while testing the application.
- List of test cases included in testing the application.
- List of features to be tested.
- The sort of approach to be used while testing the software.
- List of deliverables that need to be tested.
- The resources allocated for testing the application.
- Any risks involved during the testing process.
- A schedule of tasks and milestones to be achieved.

7. Define the following terms

i. **Error**

When the system produces an outcome, which is not the expected one or a consequence of a particular action, operation, or course, is known as error.
Error or mistake leads to a defect and usually raises due to various reasons. It may be system specification issue or design issue or coding issue, which leads to a defect. Error leads to defects and if the defect uncovered by QA leads to Failure.

ii. **Defect**

A Software DEFECT / BUG is a condition in a software product which does not meet a software requirement. In other words, a defect is an error in coding or logic that causes a program to malfunction or to produce incorrect/unexpected results.

iii. **Failures**

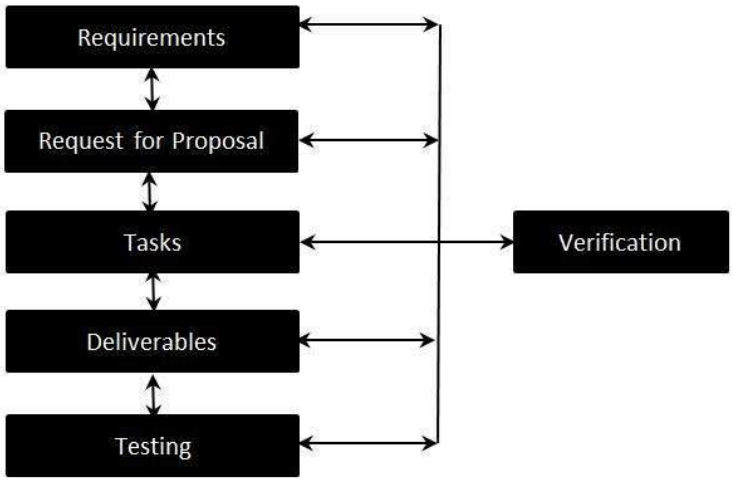
A failure is the inability of a software system or component to perform its required functions within specified performance requirements. A failure is a consequence of defect. The software is said to be a failure when it fails to perform in the real environment. The failure is also caused due to faults in the hardware system. The environment conditions in which the software is expected to perform can cause the failure. The errors lead to defects and defects lead to failure of the software.

8. Short note on requirement Traceability Matrix

Requirements tracing, a process of documenting the links between the requirements and the work products developed to implement and verify those requirements. The RTM captures all requirements and their traceability in a single document delivered at the conclusion of the life cycle.

The Matrix is created at the very beginning of a project as it forms the basis of the project's scope and deliverables that will be produced.

The Matrix is bi-directional, as it tracks the requirement forward by examining the output of the deliverables and backward by looking at the business requirement that was specified for a particular feature of the product.



Parameters of RTM are:

- Requirement ID
- Risks
- Requirement Type
- Requirement Description
- Trace to Design Specification
- Unit Test Cases
- Integration Test Cases
- System Test Cases
- User Acceptance Test Cases
- Trace to Test Script

9. What Skills required by a Tester.

Following are the basic skills required to be tester

- Written and verbal communication
- Problem solving
- Active listening
- Observations
- Testing skills ie.
  Concept of testing, levels of testing, Techniques for verification and validation, selection and using of testing tool, knowledge of testing stand.
- Client's perspective approach
- Creativity and innovation
- Continuous improvement
- Prioritizing of workflow

10. Why independent testing team is required in organizations?

Independent testing corresponds to an independent team, who involve in testing activities other than developer to avoid author bias and is often more effective at finding defects and failures.

The following list shows the increasing levels of independence for testing:

- Testing done by developer himself
- Independent testers ceded to the development team
- Independent Testing Team within Organization
- Independent Testers of different Organization
- Outsourced test team members of other organization

Benefits:

- The tester sees each defect in a neutral perspective
- The tester is totally unbiased
- The tester sees what has been built rather than what the developer thought
- The tester makes no assumptions regarding quality

Disadvantages:

- Isolation from the development team can sometimes lead to outdated documentation reference.
- The independent test execution is normally the last stage and affected by any delays earlier in the process.
- Developers might be irresponsible for quality as they might assume that independent testing team is there to find the issues within the system

- Independent testing can sometimes act as a hindrance to communication.

11. What is impact of defect in different phases of software development?

Defect Severity is the impact that a defect has on either the development or execution of any program. It is the degree of impact that a defect has, on the application.

The higher the degree of impact or severity, the more detrimental the error will be. In short, defect severity during the process of software testing depends on how important the tested function is and whether it is meeting the defined requirements of the client or not.

Defects play an important role in the Software Development Life Cycle (SDLC) and can impact the performance and the functionality of the product. It is with the assistance of defect severity that the QA team can resolve the critical defects & issues in the system and preparing a defect-free software.

Other features that make defect severity an integral part of STLC are:

- The severity level of defect indicates the potential business impact of the ends user.
- Defect severity indicates the quality of the software under test (SUT).
- This information is used to make the release decision based on the number of defects and their severity.
- It is related to the technical aspect of the product.
- The classification of a defect based on its impact on the functionality and the operation of the system is known as severity.
- If a functionality is blocked or if it functions incorrectly, it is allotted the highest defect severity.
- It assists the testing teams to determine the efficiency of the testing process.
- Helps the Quality Assurance team determine the defect priority and severity, which enables them to test higher priority defects first.
- Increases the efficiency of bug tracking, which further improves the quality of the product.
- By defining the defect severity we can identify the aspects of the software that functions incorrectly.

12. What is bug tracking, bug fixing & bug verification?

Bug tracking
Bug tracking is the process of logging and monitoring bugs or errors during software testing. It is also referred to as defect tracking or issue tracking. Large systems may have hundreds or thousands of defects. Each needs to be evaluated, monitored and prioritized for debugging. In some cases, bugs may need to be tracked over a long period of time.

Bug Fixing
Bug fixing is a change to a system or product designed to handle a programming bug/glitch. Many different types of programming bugs that create errors with system implementation may require specific bug fixes that are successfully resolved by a development or other IT team.

Bug Verification

Bug verification is done to verify that bugs are properly fixed and to check that the fixes don’t yield any new issues. This is a very important part of the QA work and one of the more challenging areas of community involvement.