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

Software Testing and Quality Assurance

Assignment 1

Unit 1: Software Testing Introduction

1. *Define the term Quality. What is bug; defect, error & failure give an example of each?*

Quality is the degree to which a component, system or process meets specified requirements and clients' needs and expectations.

In terms of Software Quality, it's the totality of functionality and features of a software product that bear on its ability to satisfy stated or implied needs.

Bug
A bug is an error which would have been introduced in the due course of the software development life cycle. The most common sources of Bugs are detailed below

- Ambiguous Requirements
- Programming Errors
- Unachievable Deadlines

Every bug not only breaks the functionality it occurs in but also has probabilities of causing a ripple effect in other areas of the application. This ripple effect needs to be evaluated when fixing any bug. Lack of foresight in anticipating such issues can cause serious problems and an increase in bug count.

Defect
A software bug arises when the expected result doesn't match with the actual results. It can also be error, flaw, failure, or fault in a computer program. Most bugs arise from mistakes and errors made by developers, architects. Some common types of defects are Arithmetic Defects Logical Defects, Syntax Defects, Multithreading Defects, Interface Defects and Performance Defects.

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.

Failure
Under certain circumstances, the product may produce wrong results. It is defined as the deviation of the delivered service from compliance with the specification.

Not all the defects result in failure as defects in dead code do not cause failure. Reason for failure could be due to Environmental conditions, which might cause hardware failures or change in any of the environmental variables; Human Error while interacting with the software by keying in wrong inputs or Failures may occur if the user tries to perform some operation with intention of breaking the system.

2. *Write short note on TQM*

Total Quality Management (TQM) can be defined as a management technique for improving processes, products, services, and the other approaches associated with the product. It focusses on the entire business and NOT just on a project or process.

Elements of TQM are following below:

- Root Cause Analysis
- Customer-focused
- Active Employee Participation
- Process-oriented
- Internal and External self-Assessment
- Continuous improvement
- Making Well Informed Decisions
- Effective Communication

Quality Control Tools used in TQM are:

- Cause - Effect Diagram
- Checklists
- Histogram
- Graphs
- Pareto Charts
- Tree diagram
- Arrow diagram

3. *Explain the term Customer as a King.*

- External Customer - outside the organization, who pay for the project
    - End-user customers
    - Manufacturer (OEM) for suppliers.
- Internal Customer - people within your organization who receive your work
- In many situations, service providers have multiple clients/ customers and therefore find it useful to identify “core customers”
- A customer’s perception is their reality.
- It is easier to keep your customers happy than attract new ones.

- Complaints spread like wildfire on the internet.
- Without customers we do not have a business.
- Brands win or lose by how well they wow customer.
- TQM’s Customer Approach
    - “the customer defines quality.”
    - “the customer is always right.”
    - “the customer always comes first.”
    - “quality begins & ends with the customer.”

4. List and explain Problem Solving Techniques.

- Techniques indicate more about a process used in measurement, analysis and decision-making during problem solving.
- Improving quality of products and services offered to customers requires methods and techniques of solving problems associated with development and processes used during their lifecycle.
- An organization must use metric approach of process improvement because it needs to make quantitative measurements.
- These measurements can be accomplished by both qualitative and quantitative methods, but problem definition becomes easier when we put some measures.
- Qualitative problem solving refers to understanding a problem solution using only qualitative indexes such as high, medium, low etc. depending on the something is improving from present status and so forth.
- Quantitative problem solving requires specification of exact measures of exact measures in numerical terms such as the cost of the 32.5% during last quarter or time required to one product is reduced by 32 minutes.
- It must follow define, measure, monitor, and control and improve cycle.

5. Explain Problem solving Software Tools.

- Tools are an organizations analytical asset that assist in understanding a problem through data and try to indicate possible solutions.
- Quality tools applied for solving problems face by projects and functional terms while improving quality in organization.
- Tools may be hardware/software and physical/logical tools.

Advantages of Using Software Tools for analysis and decision making

- Accuracy and speed of the tools is much higher compared to performing all transactions and calculations manually.
- Decision support offered of the tool is independent of personal skills and there is least variation from instance to instance.
- Tools can be integrated with other systems to provide a systematic and highly integrated means of solving problems.

Disadvantages of Using Computer Tools for Analysis and Decision Making

- Tools may mean more cost and time to learn and implement.

6. Explain Software development process

The Software Development Life (SDLC) Cycle explains the various stages of a software cycle and the structure in which these stages are carried out. The result produced from each stage is implemented in the next stage of the software life cycle. Requirements are converted into design and the design is used to develop the code. The final testing stage authenticates the results of the implementation stage by measuring it across the requirements. There are mainly five stages in the SDLC:

Requirement Analysis

The requirements of the software are determined at this stage. Discussions are held between the various stake holders, managers, and users to find out what the particular software will be used for. Who will use it and how will they be using it? Information regarding what kind of input is required and what output is expected is collected during this stage. Once the information is collected, it is analysed to see if the requirements can be incorporated into the software that is to be developed. After which, a 'Requirement Specification' document is developed to be used as a guide for the next stage.

Design

Here, the software and system design are developed according to the instructions provided in the 'Requirement Specification' document. The design stage establishes what hardware and what system requirements are needed as well as the entire system architecture. The results from this stage are used as input for the next one.

Implementation & Coding

In this stage, the actual coding is done, and the code is produced based on the design specifications. This is the most critical and also the longest stage in the SDLC.

Testing

After the development of the code, it is tested to see if it meets all the requirements that were determined in the first stage. Various kinds of testing such as system testing, unit testing, acceptance testing, and integration testing are carried out.

Maintenance

This is the final stage, where the finished software is delivered to the customer. The real problems are identified once the customer begins use. These problems are addressed from time to time as they crop up.

7. Identify problematic area of SDLC

Problematic areas of SDLC are as follows:

- No time allocated for good design and architecture
- Code becomes unreadable and unmaintainable

- Code is disorganized limiting amount of software engineers and/or changes to be active at any time.
- Makes full and patch releases difficult or impossible without severe downtime or refactoring of code.
- Simple configuration or system administration requires a code change not variable/parameter change.
- Makes change extremely difficult – not scalable or object orientated.
- Duplication of solutions i.e. not sharing such as application and hardware servers
- Makes releases extremely difficult i.e. No thought how to release the project – to many manual processes in a release. Releases take several hours and even days. Build an object too large to get through the door.
- Lack of necessary documentation or too much unnecessary documentation
- No or very little enterprise development, testing, or security standards
- Non-technical staff getting involved in making technical decisions causes the two most common features in struggling and/or failing technology projects:
- Pressure from business to implement change fast. Management scheduling without understanding impact and risk
    - Inability of IT to explain and the business to understand the intricate nature of technology projects.

8. Describe Pillars of Quality Management System

The 5 Pillars of a Modern Quality Management System include:

- Integrated Processes
- System Flexibility
- Monitoring and Management
- Compliance Enablement
- Culture of Quality and Compliance

Pillar 1: Process Integration

While most companies execute on these functions, the degree of integration and automation varies greatly from company to company. The concern of the first pillar is how well these processes are integrated together. For example, if the Customer Complaint process is completely separated from the Corrective Action process, then there is room for improvement.

Pillar 2: System Flexibility and Extensibility

Unfortunately, no one can predict what the future holds. Being able to adapt and continuously innovate is extremely important. In summary, the second pillar is all about connectivity, expandability, and configurability.

Pillar 3: Centralized Monitoring and Management

A best of breed enterprise quality management system should provide the following capabilities:

- Alerts & Notifications
- Reporting
- Measurements & KPIs
- Management Dashboards

Pillar 4: Compliance Enablement

Almost all industries have regulatory standards and requirements. A modern enterprise quality management system should have compliance built into the system. Compliance should not be an afterthought.

Pillar 5: Culture of Quality & Compliance

Establishing a culture of quality and compliance, perhaps the most important quality management pillar, shows a commitment from the top that flows down throughout the rest of the organization. A quality system, no matter who well designed, cannot work if there is no visible management buy-in.

9. List down characteristics of Software.

The following are the **characteristics of software**:

i. Software does not wear out
ii. Software is not manufactured
iii. Usability of Software
iv. Reusability of components
v. Flexibility of software
vi. Maintainability of software
vii. Portability of software
viii. Reliability of Software

Software does not wear out
Different things like clothes, shoes, ornaments do wear out after some time. But software once created never wears out. It can be used for as long as needed and in case of need for any updating, required changes can be made in the same software and then it can be used further with updated features.

Software is not manufactured
Software is not manufactured but is developed. So, it does not require any raw material for its development.

Usability of Software
The usability of the software is the simplicity of the software in terms of the user. The easier the software is to use for the user, the more is the usability of the software as more number of people will now be able to use it and also due to the ease will use it more willingly.

Reusability of components
As the software never wears out, neither do its components, i.e. code segments. So, if any particular segment of code is required in some other software, we can reuse the existing code form the software in which it is already present. This reduced our work and also saves time and money.

Flexibility of software
A software is flexible. What this means is that we can make necessary changes in our software in the future according to the need of that time and then can use the same software then also.

Maintainability of software
Every software is maintainable. This means that if any errors or bugs appear in the software, then they can be fixed.

Portability of software
Portability of the software means that we can transfer our software from one platform to another that too with ease. Due to this, the sharing of the software among the developers and other members can be done flexibly.

Reliability of Software:
This is the ability of the software to provide the desired functionalities under every condition. This means that our software should work properly in each condition.

10. *Define constraints of software Product quality Assessment*

Requirement specification are made by business analyst and system analyst. Tester may or may not have direct access to the customer and may get information though requirement statements, queries answered etc. either from customer or business analyst. There are few limitations of product quality assessment in this.

- Software is virtual in nature. Software products cannot be touched or heard.
- There is huge communication gap between users of software and developers/testers of the product.
- Software product is unique in nature. Similarities between any two products are superficial ones.
- All aspects of software cannot be tested fully as member of permutations and combinations for testing all possibilities tend to infinity.
- A software program in the same way every time when it is executing some instruction.

11. *Differentiate between Software tools and Techniques*

Software Tools	Solving Techniques
Tools are an organizations analytical asset that assist in understanding a problem through data and try to indicate possible solutions.	Techniques indicate more about a process used in measurement, analysis and decision-making during problem solving.
Quality tools are applied for solving problems faced by projects and functional terms while improving quality in organization.	Improving quality of products and services offered to customers requires methods and techniques of solving problems associated with development and processes used during their lifecycle.
Tools may be hardware/software and physical/logical tools.	It can be accomplished by both qualitative and quantitative methods, but problem definition becomes easier when we put some measures.
Accuracy and speed of the tools is much higher compared to performing all transactions and calculations manually.	Qualitative problem solving refers to understanding a problem solution using only qualitative indexes such as high, medium, low etc. depending on the something is improving from present status and so forth.

Decision support offered of the tool is independent of personal skills and there is least variation from instance to instance.	Quantitative problem solving requires specification of exact measures of exact measures in numerical terms such as the cost of the 32.5% during last quarter or time required to one product is reduced by 32 minutes.
Tools may mean more cost and time to learn and implement.	It may follow define, measure, monitor and control and improve cycle.

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
A bug fix is, simply the fix to a bug, that is, the set of modifications to the software system that would correct the flaw.

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.

13. What is software testing process? Why do we have to test the software? Explain the reasons with respect to quality view, financial aspect, customer's suppliers, and process.

- Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. In simple words, testing is executing a system to identify any gaps, errors, or missing requirements in contrary to the actual requirements.
- The testing is important since it discovers defects/bugs before the delivery to the client, which guarantees the quality of the software. It makes the software more reliable and easier to use. Thoroughly tested software ensures reliable and high-performance software operation.
- Testing has many benefits and one of the most important ones is cost-effectiveness. Having testing in your project can save money in the long run. Software development consists of many stages and if bugs are caught in the earlier stages it costs much less to fix them. That is why it's important to get testing done as soon as possible. Another important point to add is security. This is probably the most sensitive and yet most vulnerable part.
- There have been many situations where user information has been stolen or hackers have gotten to it and used it for their benefit. That is the reason people are looking for trusted products that they can rely on.

- Products always serve users in some ways, so it’s very important that it brings the value it promises, hence it should work properly to ensure great customer experience. Development of an app, for example, has many processes included and testing gets a glimpse of every bit – it checks if the apps graphics are aligned properly, tests the main functionality, checks if menus are intuitive, etc. After developers fix issues, sometimes another issue may appear unexpectedly somewhere else, that’s just how testing goes sometimes, so it’s great to find those issues to be resolved and be a part of quality product being delivered to marketplace.
- Reasons why apps and software should be tested is to bring the best user experience possible. Being the best product in this saturated market will help you gain trustworthy clients which will have great long-term effects. Once users will have amazing customer experience they will, without a doubt, tell their friends and word to mouth will make it advertise itself, but this works both ways.

14. What is software quality? what is mean by quality attribute of the software? Explain continual improvement cycle with neat labelled diagram of plan-Do check-Act (PDCA) framework

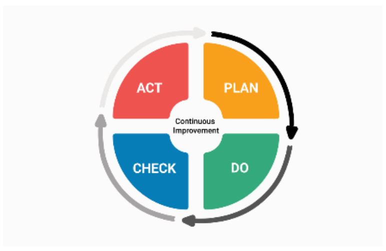
Software quality is defined as a field of study and practice that describes the desirable attributes of software products. There are two main approaches to software quality: defect management and quality attributes.

This approach to software quality is best exemplified by fixed quality. This standard describes a hierarchy of eight quality characteristics, each composed of sub-characteristics:

1. Functional suitability
2. Reliability
3. Operability
4. Performance efficiency
5. Security
6. Compatibility
7. Maintainability
8. Transferability

he most widely used tools for the continuous improvement model is a four-step quality assurance method—the plan-do-check-act (PDCA) cycle:

- **Plan:** Identify an opportunity and plan for change.
- **Do:** Implement the change on a small scale.
- **Check:** Use data to analyse the results of the change and determine whether it made a difference.
- **Act:** If the change was successful, implement it on a wider scale and continuously assess your results. If the change did not work, begin the cycle again.



Plan-Do-Check-Act Diagram