
# Debre Markos University
# Institute of Technology
# School of Computing, Software Eng A/program

## Software Testing and Quality Assurance(SEng4051)

## Chapter One

## SQA Concepts

# Contents

- Introduction
- Quality
- Detection Vs Prevention
- Verification and Validation
- Testing

# Introduction

**Quality?**

- Quality is an intangible concept.
- The terms **good quality**, **poor quality** are used in our everyday life to tell how **good** or **bad** a product functions.
- **Quality** is a complex concept it means different things to different people, and *it is highly context dependent.*
- To be able to capture the quality concept, it is important to study quality from a broader perspective. This is because the concept of quality predates software development.

# Views of Quality

## Transcendental View

- In the **transcendental view**, quality is something that can be recognized through experience but is not defined in some tractable form.
- Quality is viewed to be something ideal, which is too complex to lend itself to be precisely defined. However, a good-quality object stands out, and it is easily recognized.

## User's View

- It perceives quality as fitness for purpose. According to this view, while evaluating the quality of a product, one must ask the key question: "**Does the product satisfy user needs and expectations?**"

# Cont...

**Manufacturing View :**

- Here quality is understood as conformance to the specifications.
- The quality level of a product is determined by the extent to which the product meets its specifications

**Product View :**

- Quality is viewed as tied to the inherent characteristics of the product.
- A product's internal qualities determine its external qualities.
- Quality is that high degree of modularity, which is an internal property, makes a software testable and maintainable.

**Value - Based View :**

- Quality, in this perspective, depends on the amount the customer is willing to pay for it.

# What is Software?

**Based on IEEE definition**

- **Software** is: computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system.
- ISO definition lists out the following components of software:
  - Computer programs (the "code")
  - Procedures
  - Documentation
  - Data necessary for operating the software system

# Software Quality

**Pressman's definition:**

- Conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software.

**IEEE definitions:**

- The degree to which a system, component, or process meets specified requirements.
- The degree to which a system, component, or process meets customer or user needs or expectations.

# Detection vs Prevention

- **Quality Assurance (QA):** Is a set of activities for ensuring quality in the process by which products are developed.
- **QA** focuses on improving software development process.
- **Quality Control (QC)** - Is a set of activities for ensuring quality in products.
- **QC** does not deal with the processes used to create a product; rather it examines the quality of the "end products" and the final outcome

# Cont...

| Quality Assurance | Quality Control |
|---|---|
| Aim is to prevent the defect | Aim is to identify and improve the defects |
| Proactive approach | Reactive approach |
| Does not involve executing the program | Involves executing the program |
| About engineering the process | Examine the product |
| Is the technique of managing quality | Is a method to verify quality |
| Is responsible for full software development life cycle | Responsible for software testing life cycle |
| All team members are responsible for QA | Testing team is responsible for QC |

# Software Verification Vs Software Validation

- **Verification** helps in ensuring if the software is being developed in the right way.
- The focus of verification is on the quality of software that is being developed, whether it follows all the standards or not, is it well engineered?
- **Validation** on the other hand helps in building the right software.
- While carrying out validation we look at whether the software is in line with customer's requirement or not.

# Cont...

| Verification | Validation |
|---|---|
| Verification includes checking documents, design, code and program | It is a dynamic mechanism of testing and validating the actual product |
| It does not involve executing the code | It always involves executing the code |
| Verification uses methods like reviews, walkthroughs, inspections and desk-checking etc. | It uses methods like Black Box Testing, White Box Testing and non-functional testing |
| It checks whether the software conforms to standards or not | It checks whether software meets the requirements and expectations of customers or not |
| It finds bugs early in the development cycle | It can find bugs that the verification process can not catch |
| The target is a software architecture, specification, complete design, high level and data base design etc. | The target is an actual product |
| QA team does verification by using the SRS, design, etc. documents. | With the involvement of testing team validation is executed on software code. |
| It comes before validation | It comes after verification |

# Software Testing

- An activity of checking whether the actual results match the expected results and to ensure that the software is defect free.
- The process of verifying and validating a software application to check whether it is working as expected or not.
- The intent is to find defects and improve the product quality.
- Software testing also helps to identify errors, gaps or missing requirements in contrary to the actual requirements.
- It can be done either manually or using automated tools.

# Objective of Software testing

- To find any defects or bugs that may have been created when the software was being developed
- To increase confidence in the quality of the software
- To prevent defects in the final product
- To ensure the end product meets customer requirements as well as the company specifications
- To provide customers with a quality product and increase their confidence in the company

# Defect, Error, Bug, Failure

A mistake in coding is called error;

**an error** found by a tester is called defect;

**a defect** accepted by a development team is called bug;

if a build* does not meet the requirements, it is a failure.

# Cont...

- **An Error** is a mistake made in the code due to which compilation or execution fails.
- **A Defect** is a deviation between the actual and expected output.
- **A bug** refers to defects which means that the software product or the application is not working as per the adhered requirements set.
- **Failure** is the accumulation of several defects that ultimately lead to Software failure and results in the loss of information in critical modules thereby making the system unresponsive.

# Causes of Software Errors

**The nine causes of software errors are:**

- Faulty requirements definition
- Client-developer communication failures
- Deliberate deviations from software requirements
- Logical design errors
- Coding errors
- Non-compliance with documentation and coding instructions
- Shortcomings of the testing process
- User interface and procedure errors
- Documentation errors

**Faulty Requirements Definition**

- Usually considered the root cause of software errors
- Incorrect requirement definitions
  - Simply stated, 'wrong' definitions (formulas, etc.)
- Incomplete definitions
  - Unclear or implied requirements
- Missing requirements
  - Just flat-out 'missing.' (e.g. Program Element Code)
- Inclusion of unneeded requirements
  - (many projects have gone amuck for including far too many requirements that will never be used.
  - Impacts budgets, complexity, development time, …

# Cont...

**Client-developer communication failures**

- Misunderstanding of instructions in requirements documentation (written / graphical instructions)
- Misunderstanding of written changes during development.
- Misunderstanding of oral changes during development.
- Lack of attention
- ✓ to client messages by developers dealing with requirement changes and
- ✓ to client responses by clients to developer questions
- Very often, these very talented individuals come from different planets, it seems.
- ✓ Clients represent the users; developers represent a different mind set entirely some times!

# Cont...

**Deliberate deviations from software requirements**

- Developer reuses previous / similar work to save time.
- Often reused code needs modification which it may contain contain unneeded / unusable extraneous code.
- Book suggests developer(s) may overtly omit functionality due to time / budget pressures.
  - ✓ Another BAD choice; System testing will uncover these problems to everyone's dismay!
  - ✓ I have never seen this done intentionally – but understand it!
- Developer inserting unapproved 'enhancements' (perfective coding; a slick new sort / search ); may also ignore some seemingly minor features, which sometimes are quite major.
  - ✓ Have seen this and it too causes problems and embarrassment during reviews.

## Logical design errors

- Definitions that represent software requirements by means of erroneous algorithms.
  - Wrong formulas
  - Wrong Decision Logic Tables
  - Incorrect descriptions in text
- Process definitions: procedures specified by systems analyst not accurately reflecting the real business process.
  - Note: all errors are not necessarily software errors.
  - This seems like a procedural error, and likely not a part of the software system. But they are errors nonetheless!
- Erroneous Definition of Boundary Condition – a common source of errors.
  - The "absolutes" like 'no more than' "fewer than" "n times or more" "the first time" etc

# Cont...

**Coding errors**

- Too many to try to list.
  - Syntax errors (grammatical errors)
  - Logic errors (program runs; results wrong)
  - Run-time errors (crash during execution)

# Cont...

**Non-compliance w/documentation & coding instructions**

- Non-compliance with published templates (structure)
- Non-compliance with coding standards (attribute names)
- Size of program;
  - Other programs must be able to run in environment!
  - Data Elements and Code.
  - Required documentation manuals and operating instructions
- SQA Team: testing not only execution software but coding standards; manuals, messages displayed; resources needed; resources named (file names, program names).

## Shortcomings of the Testing Process

- Likely the part of the development process cut short most frequently!
- Incomplete test plans
  - Parts of application not tested or tested thoroughly!
- Failure to document, report detected errors and faults
  - So many levels of testing….we will cover.
- Failure to quickly correct detected faults due to unclear indications that there 'was' a fault
- Failure to fix the errors due to time constraints
  - Many philosophies here depending on severity of the error.

# Cont...

**User interface and procedure errors**

- Missing or wrong functionality
- Spelling, actual and context error

**Documentation errors**

- Errors in the design documents
  - Trouble for subsequent redesign and reuse
- Errors in the documentation within the software for the User Manuals
- Errors in on-line help, if available.
- Listing of non-existing software functions
  - Planned early but dropped; remain in documentation!

# Software Testing Principles

- A principle of software testing refers to the brief mentioned and proven concepts which guide testing professionals during software testing process.

**Principle 1.Testing shows the presence of bugs**

- Testing shows the defects but cannot prove that there are no defects. Meaning that the testing team cannot say that the product is 100% defect-free. ◦
- Testing reduces the number of undiscovered defects in the application. ◦
- Therefore, it is important to design test cases which find as many defects as possible.

## Principle 2. Exhaustive testing is impossible

- It is impossible to test all possible combinations of input cases and data.
  - For instance, consider there are 15 fields in one screen which contain 5 possible values. To test all combinations, you would need 515 = (30,517,578,125 tests. But, project timescales would never allow testing a large number of combinations.
- For this reason, severity and priority are used to concentrate on the most important aspects to test.

## Principle 3. Early testing

- The sooner we start the testing activities, the better we can utilize the available time.
- As soon as the initial products, such as the requirement or design documents are available, we can start testing.
- It is common for the testing phase to get squeezed at the end of the development lifecycle, i.e. when development has finished, so by starting testing early, we can prepare testing for each level of the development lifecycle.
- When defects are found earlier in the lifecycle, they are much easier and cheaper to fix.
- The main advantage of early testing is testers can easily detect errors and help in each level of development with fewer costs and efforts.

**Principle 4. Defect clustering**

- During testing, it can be observed that most of the reported defects are related to small number of modules within a system.
- This is the application of the Pareto Principle to software testing: approximately 80% of the problems are found in 20% of the modules.
- The Pareto Principle defines "the vital few, the trivial many" Bugs are uneven in frequency – a vital few contribute the majority of the program failures. Fix these first.

## Principle 5. The pesticide paradox

- If you keep running the same set of tests over and over again, no more new defects will be discovered by those test cases.
- Because as the system evolves, many of the previously reported defects will have been fixed and the old test cases do not apply anymore.
- Anytime a fault is fixed or a new functionality added, we need to do regression testing to make sure the new changed software has not broken any other part of the software.
- However, those regression test cases also need to change to reflect the changes made in the software to be applicable and hopefully find new defects.

**Principle 6. Testing is context dependent**

- Different methods, techniques and types of testing are related to the type and nature of the application.
- For example, a software application in a medical device needs more testing than a game software. More importantly a medical device software requires risk based testing, be compliant with medical industry regulators and possibly specific test design techniques.
- Similarly, a very popular website, needs to go through rigorous performance testing as well as functionality testing to make sure the performance is not affected by the load on the servers.

## Principle 7. Absence of errors fallacy

- Just because testing didn't find any defects in the software, it doesn't mean that the software is ready to be shipped.
- Were the executed tests really designed to catch the most defects? or
- Were they designed to see if the software matched the user's requirements?
- There are many other factors to be considered before making a decision to ship the software.

# What is SQA?

- The degree to which a system, component, or process meets specified requirements.
- The degree to which a system, component or process meets customer or user needs or expectations.
- SQA encompasses the entire software development process such as
  - software requirements
  - software design
  - Coding
  - source code control
  - code reviews
  - change management
  - configuration management
  - release management

# THE END

# THANK YOU!!!