SNJB's KBJ College of Engineering

Chandwad–423101 (Nashik)

Department of Information Technology

Subject : Software Testing & Quality Assurance (STQA) of BE 2015 Pattern

UNIT – I
SOFTWARE TESTING BASICS

Testing as an Engineering Activity

- Exciting time to be a software developer
- Software systems are becoming more challenging to build
- Playing an increasingly important role in society
- People with software development skills are in demand
- New methods, techniques, and tools are becoming available to support development and maintenance tasks

Testing as an Engineering Activity

- As software plays an important role in our lives both economically and socially, there is pressure for software professionals to focus on quality issues
- Poor quality software that can cause loss of life or property is no longer acceptable to society
- Failures can result in catastrophic losses
- Demand software development staffs with interest and training in the areas of software product and process quality

Testing as an Engineering Activity

- Highly qualified staff ensure that software products are
  - Built on time
  - Within budget
  - And are of highest quality with respect to reliability, correctness, usability and ability to meet all user requirements

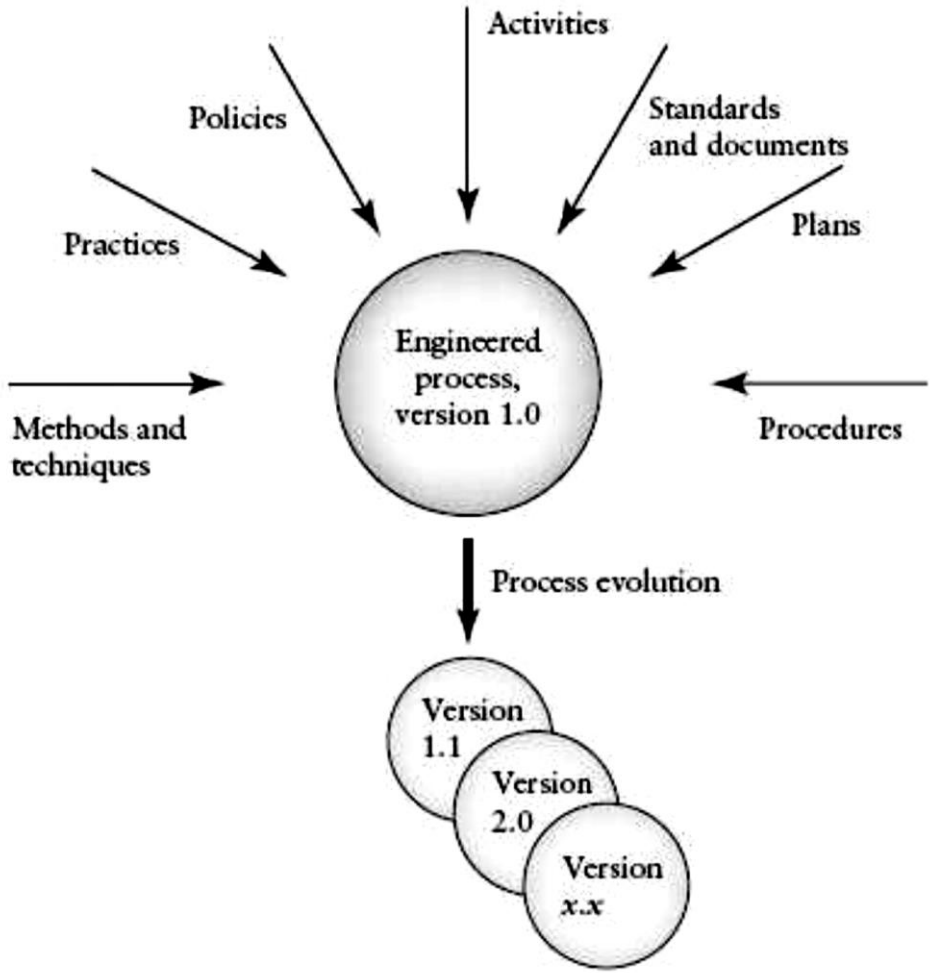
Testing as an Engineering Activity

- Using an engineering approach to software development indicates that:
  - Development process is well understood
  - Projects are planned
  - Life cycle models are defined and followed
  - Standards are in place for product and process
  - Components are reused
  - Validation and verification processes play a key role in quality determination
  - Engineers have proper education, training, and certification

Role of Process in Software Quality

Process, in software engineering domain, is set of methods, practices, standards, documents, activities, policies, and procedures that software engineers use to develop and maintain a software system and its associated artifacts, such as project and test plans, design documents, code, and manuals.

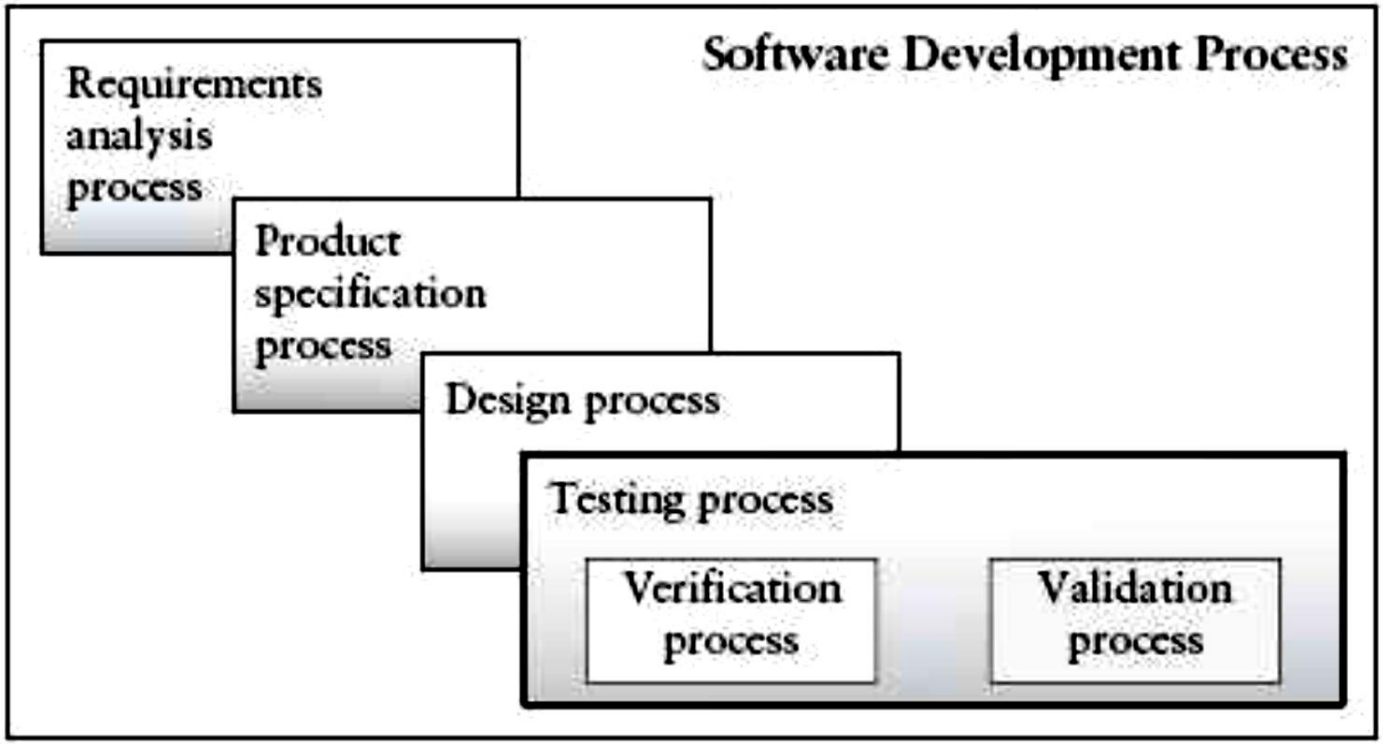
Role of Process in Software Quality



Testing as a Process

- Software development process is series of phases, procedures & steps that result in production of software product
- Embedded within software development process are several other processes including testing
- Testing itself is related to two other processes called verification and validation

Testing as a Process



Testing as a Process

Validation is the process of evaluating a software system or component during, or at the end of, the development cycle in order to determine whether it satisfies specified requirements.

Verification is the process of evaluating a software system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase

Testing as a Process

Testing is generally described as a group of procedures carried out to evaluate some aspect of a piece of software.

Testing can be described as a process used for revealing defects in software, and for establishing that the software has attained a specified degree of quality with respect to selected attributes.

Testing as a Process

- Testing and debugging, or fault localization, are two very different activities
- Debugging process begins *after* testing has been carried out
- And tester has noted that software is not behaving as specified

Debugging or fault localization is the process of (1) locating the fault or defect, (2) repairing the code, and (3) retesting the code

Basic Definitions

- Error - Mistake, misconception, misunderstanding on the part of a software developer
- Fault (Defect/Bug) - Introduced into the software as a result of an error
- Failure - Inability of software system or component to perform its required functions within specified performance requirements

Basic Definitions

- Test case - Test case in practical sense is test-related item which contains the following information:
  - *A Set of Test Inputs*
  - *Execution Conditions*
  - *Expected Outputs*
- Test - Group of related test cases, or a group of related test cases and test procedures
- Test Oracle - Document, or piece of software that allows testers to determine whether a test has been passed or failed

Basic Definitions

- Test bed - Environment that contains all hardware and software needed to test software component or software system
- Software Quality - Relates to degree to which system, system component, or process meets
  - specified requirements
  - customer or user needs, or expectations

Basic Definitions

- Quality metric - Quantitative measurement of degree to which item possesses given quality attribute
- Some examples of quality attributes are as follows:
  - Correctness
  - Reliability
  - Usability
  - Integrity
  - Portability
  - Maintainability
  - Interoperability

Basic Definitions

- Software Quality Assurance Group
  - Deals with quality issues
  - Serves as the customers‘ representative and advocate
  - Their responsibility is to look after customers‘ interests
  - Team of people with necessary training and skills
- Review - Group meeting whose purpose is to evaluate a software artifact or a set of software artifacts

Software Testing Principles

- Principles play an important role in all engineering disciplines
- Usually introduced as part of an educational background in each branch of engineering
- Testing principles are important to test specialists/ engineers
- Because they provide foundation for developing testing knowledge and acquiring testing skills

Software Testing Principles

- A principle can be defined as:
  - General or fundamental, law, doctrine, or assumption
  - Rule or code of conduct
  - Laws or facts of nature underlying the working of an artificial device
- Software engineering principles refer to
  - Laws, rules, or doctrines that relate to software systems
  - How to build them
  - How they behave

Software Testing Principles

- Principles may also refer to rules or codes of conduct relating to professionals who design, develop, test, and maintain software systems
- Glenford Myers has outlined such a set of *execution-based* testing principles in his pioneering book, *The Art of Software Testing*
- Some of these principles are described below

Principle 1

Testing is process of exercising a software component using a selected set of test cases, with the intent of (i) revealing defects, and (ii) evaluating quality

Principle 2

When the test objective is to detect defects, then a good test case is one that has a high probability of revealing a yet undetected defect(s)

Principle 3

Test results should be inspected meticulously

Principle 4

A test case must contain expected output or result

Principle 5

Test cases should be developed for both valid and invalid input conditions

Principle 6

The probability of existence of additional defects in a software component is proportional to the number of defects already detected in that component

Principle 7

Testing should be carried out by a group that is independent of the development group

Principle 8

Tests must be repeatable and reusable

Principle 9

Testing should be planned

Principle 10

Testing activities should be integrated into the software life cycle

Principle 11

Testing is a creative and challenging task

Tester's Role in Software Development Organization

- Tester's job is
  - To reveal defects
  - Find weak points
  - Inconsistent behavior
  - Circumstances where software doesn't work as expected
- As tester you need to be comfortable with this role
- Difficult for developers to effectively test their own code

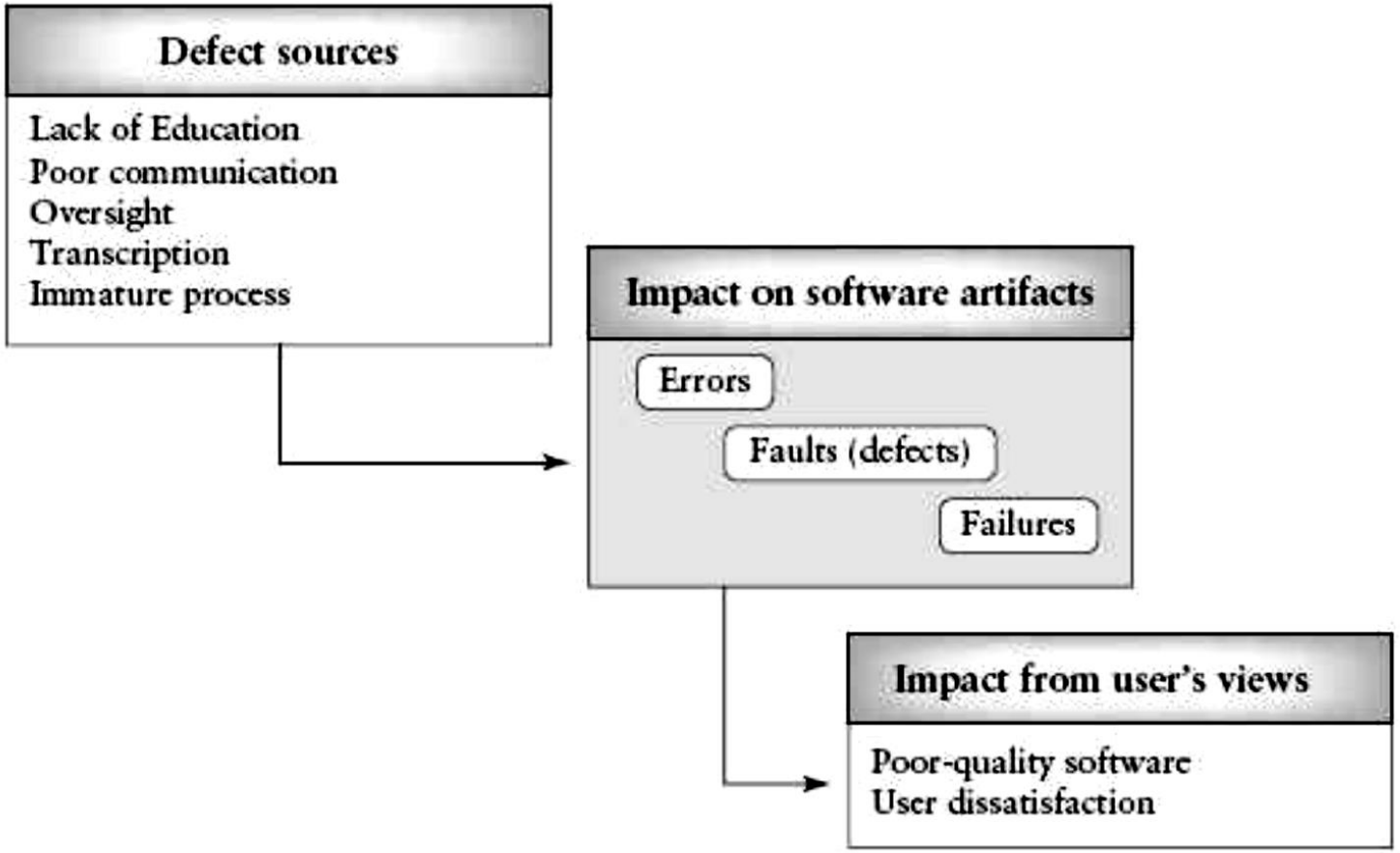
Tester's Role in Software Development Organization

- To be most effective as tester requires extensive programming experience
- Goal of tester is to work with the developers to produce high-quality software that meets the customers' requirements
- Teams of testers and developers are very common in industry
- Projects should have an appropriate developer and tester ratio

Tester's Role in Software Development Organization

- The ratio will vary depending on available resources, type of project, etc.
- Testers also need to work along side with requirements engineers
- Testers also need to work with designers to plan for integration and unit test
- In addition, test managers will need to cooperate with project managers
- Finally, testers also need to cooperate with software quality assurance staff

Origins of Defects



Defect Classes, Defect Repository and Test Design

- Defects can be classified in many ways
- It is important for an organization to adapt a single classification scheme and apply it to all projects
- No matter which classification scheme is selected
- some defects will fit into more than one class or category
- Defect types and frequency of occurrence should be used to guide test planning, and test design

Defect Classes, Defect Repository and Test Design

- Execution-based testing strategy should be selected
- Because of its strongest possibility of detecting particular types of defects
- Defects are assigned to four major classes
  - Requirements or Specifications
  - Design
  - Code
  - Testing defects

Requirements or Specifications Defect Class

- Beginning of software life cycle is critical for ensuring high quality
- Defects injected in early phases can persist and be very difficult to remove in later phases
- Many requirements documents are written using a natural language representation
- Occurrences of ambiguous, contradictory, unclear, redundant, and imprecise requirements
- Specifications are also developed using natural language representations

Requirements or Specifications Defect Class

- *Functional Description Defects* - overall description of what the product does, and how it should behave is incorrect, ambiguous, and/or incomplete
- *Feature Defects* - Due to feature descriptions that are missing, incorrect, incomplete, or superfluous
- *Feature Interaction Defects* - Due to an incorrect description of how the features should interact
- *Interface Description Defects* - Occur in description of how the target software is to interface with external software, hardware, and users

Design Defect Class

- Occur when system components, interactions between system components, interactions between the components and outside software/hardware, or users are incorrectly designed
- Defects in design of algorithms, control, logic, data elements, module interface descriptions & external software/hardware/user interface descriptions

Design Defect Class

- *Algorithmic and Processing Defects* - occur when processing steps in algorithm as described by the pseudo code are incorrect
- *Control, Logic, and Sequence Defects* - occur when logic flow in the pseudo code is not correct
- *Data Defects* - Associated with incorrect design of data structures
- *Module Interface Description Defects* - Derived from using incorrect, an incorrect number of parameters, or an incorrect ordering of parameters

Design Defect Class

- *Functional Description Defects* - Include incorrect, missing, unclear design elements
- *External Interface Description Defects* - Derived from incorrect design descriptions for external software systems, databases, and hardware devices

Coding Defect Class

- Derived from errors in implementing the code
- Closely related to design defect classes especially if pseudo code has been used for detailed design
- Come from failure to understand programming language constructs, and miscommunication with the designers
- Difficult to classify a defect as a design or as a coding defect

Coding Defect Class

- *Control, Logic and Sequence Defects* - Include incorrect expression of case statements, incorrect iteration of loops and missing paths
- *Typographical Defects* - Generally syntax errors
- *Initialization Defects* - Occur when initialization statements are omitted or are incorrect
- *Data-Flow Defects* – Due to incorrect flow of data
- *Data Defects* - By incorrect implementation of data structures

Coding Defect Class

- *Module Interface Defects* - Derived from using incorrect, an incorrect number of parameters, or an incorrect ordering of parameters
- *Code Documentation Defects* - When the code documentation does not reflect what the program actually does, or is incomplete or ambiguous
- *External Hardware, Software Interfaces Defects* - arise from problems related to system calls, links to databases, input/output sequences, memory & resource usage, interrupt & exception handling

Testing Defect Class

- Test plans, test cases, test harnesses, and test procedures can also contain defects
- Defects in test plans are best detected using review techniques
- *Test Harness Defects* - In order to test software, especially at the unit and integration levels, auxiliary code must be developed, this is called test harness.
- *Test Case Design and Test Procedure Defects*

Developer / Tester Support for Developing Defect Repository

- Important if you are member of test organization
- To illustrate to management and your colleagues the benefits of developing defect repository to store defect information
- Requirement for repository development should be part of testing and/or debugging policy statements
- Begin with development of defect classification scheme and then initiate collecting defect data from organizational projects

Developer / Tester Support for Developing Defect Repository

- Forms and templates will need to be designed to collect the data
- Defect monitoring should continue for each on-going project
- Defect data is useful for test planning
- Defect data can support debugging activities as well

Developer / Tester Support for Developing Defect Repository