

Software Engineering Testing

Introduction

What is testing?

- Software testing consists of the **dynamic verification** of the behaviour of a program on a **finite set of test cases**, suitably **selected from the usually infinite executions domain**, against the **expected behaviour**.

Source: SWEBOK, Chapter 5, Software Testing, 2004

Software Faults, Errors & Failures

- **Software Fault** : A static defect in the software
- **Software Failure** : External, incorrect behavior with respect to the requirements or other description of the expected behavior

**Faults in software are design mistakes and will always exist.
But Failures may not happen always, though there are Faults.**

Testing & Debugging

- **Testing** : Finding inputs that cause the software to fail

TC #	Description	Input/s	Expected O/P	Actual O/P	Result	Comments

- **Debugging** : The process of finding a fault given a failure

Definitions of Quality

1. Conformance to requirements (Philip Crosby) –

Producer view: characterized by:

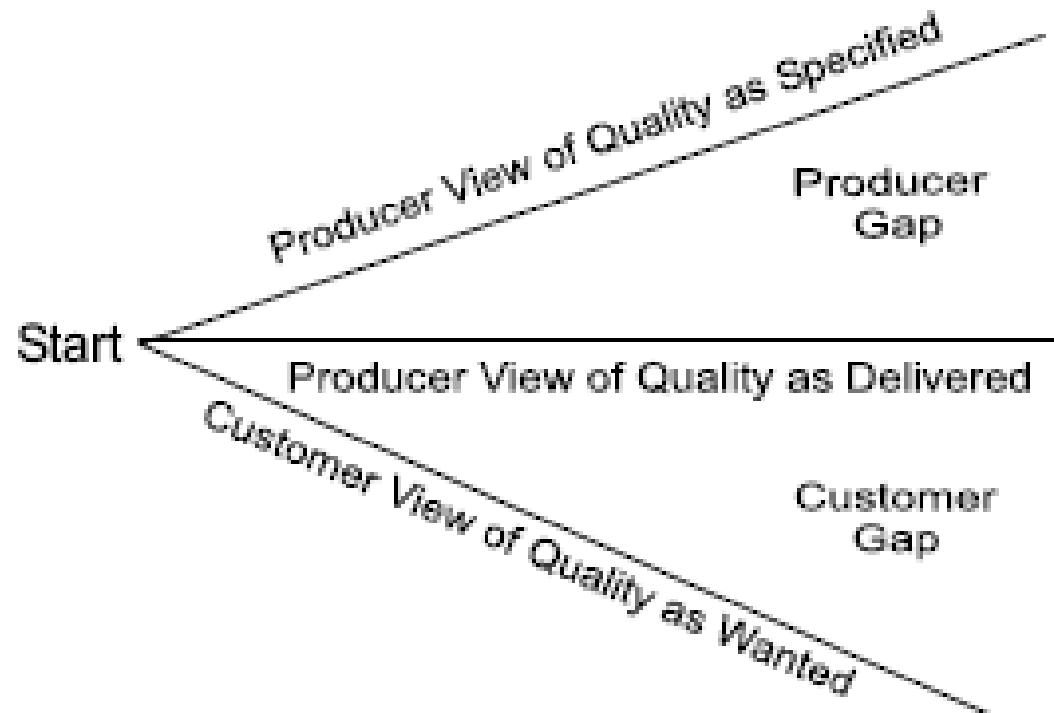
- Doing the right thing
- Doing it the right way
- Doing it right the first time
- Doing it on time without exceeding cost

2. Fit for use (Joseph Juran & Edwards Deming) –

Customer view: characterized by:

- Receiving the right product for their use
- Being satisfied that their needs have been met
- Meeting their expectations
- Being treated with integrity, courtesy and respect

Two Quality Gaps




Defects

- **Defect Prevention**
 - Preventing from defects by improving the processes.
- **Defect Detection**
 - Detecting the defects by proper appraisal mechanisms.
- **Latent Defects**
 - Defects found at the client site after deployment

The Cost of Quality

- **Prevention Costs**
 - Up-front costs for benefits that will be derived later
 - Establishing procedures, training, tools and planning
 - Spent before the product is actually built
- **Appraisal Costs**
 - Review completed products against requirements
 - Includes the cost of inspections, testing, and reviews.
 - After the product is built but before it is shipped to the user
- **Failure Costs**
 - Defects that make it to the user or to production.
 - Repairing products to make them meet requirements.
 - Cost of operating faulty products and operating a Help Desk.

Types of Test Activities

- Testing can be broken up into **four** general types of activities
 1. Test Design  1.a) Criteria-based
 2. Test Automation 1.b) Human-based
 3. Test Execution
 4. Test Evaluation

Types of Test Activities – Summary

1a.	Design	Design test values to satisfy engineering goals
	Criteria	Requires knowledge of discrete math, programming and testing
1b.	Design	Design test values from domain knowledge and intuition
	Human	Requires knowledge of domain, UI, testing
2.	Automation	Embed test values into executable scripts
		Requires knowledge of scripting
3.	Execution	Run tests on the software and record the results
		Requires very little knowledge
4.	Evaluation	Evaluate results of testing, report to developers
		Requires domain knowledge

- These four general test activities are quite different
- It is a poor use of resources to use people inappropriately

Most test teams use the same people for ALL FOUR activities !!

Terminology (V&V – by IEEE)

- Validation : The process of evaluating software at the end of software development to ensure compliance with intended usage
- Verification : The process of determining whether the products of a given phase of the software development process fulfill the requirements established during the previous phase
- IV&V stands for “*independent verification and validation*”

Terminology (V&V – by IEEE)

- Validation:
does the software system meets the user's real needs?
are we building the right software?
- Verification:
does the software system meets the requirements specifications?
are we building the software right?

Static and Dynamic Testing

- Static Testing : Testing without executing the program
 - Reviews
 - Inspections
 - Walkthroughs
- Dynamic Testing : Testing by executing the program with real inputs
 - Unit Testing
 - Component Testing
 - Integration Testing etc.

Observability and Controllability

- Software Observability : How easy it is to observe the behavior of a program in terms of its outputs, effects on the environment and other hardware and software components
 - Software that affects hardware devices, databases, or remote files have low observability
- Software Controllability : How easy it is to provide a program with the needed inputs, in terms of values, operations, and behaviors
 - Easy to control software with inputs from keyboards
 - Inputs from hardware sensors or distributed software is harder

White-box and Black-box Testing

- Black-box testing : Deriving tests from external descriptions of the software, including specifications, requirements, and design
- White-box testing : Deriving tests from the source code internals of the software, specifically including branches, individual conditions, and statements

Testing at Different Levels

- Unit testing: Test each unit individually
- Component testing: Test each component
- Integration testing: Test how modules interact with each other
- System testing: Test the overall functionality of the system
- Acceptance testing: Is the software acceptable to the user?

Test Coverage Criteria

- Test Requirements : Specific things that must be satisfied or covered during testing
- Test Criterion : A collection of rules and a process that define test requirements
- Test Set: Minimal number of Test Cases or Test Paths needed to cover Test Requirement Set

Coverage

Given a set of test requirements TR for coverage criterion C , a test set T satisfies C coverage if and only if for every test requirement tr in TR , there is at least one test t in T such that t satisfies tr

- Infeasible test requirements : test requirements that cannot be satisfied
 - No test case values exist that meet the test requirements
 - Dead code
 - Detection of infeasible test requirements is formally undecidable for most test criteria
- Thus, 100% coverage is **impossible** in practice

Fault & Failure Model

Three conditions necessary for a failure to be observed

1. Reachability : The location or locations in the program that contain the fault must be reached
2. Infection : The state of the program must be incorrect
3. Propagation : The infected state must propagate to cause some output of the program to be incorrect

Exercise

```
public int findLast (int[] x, int y)
{
    //EFFECTS: If x is empty, return -1
    //else returns the index of the last element in x that equals to y
    //if no such element exists, return -1
    if (x.length ==0)
        return -1;
    for (int i=x.length-1;i>0;i--)
    {
        If (x[i]==y)
            Return i;
    }
    Return -1
}
```

V Model In Testing (Software Testing Life Cycle)

