Introduction To Software Testing

Introduction To Software Testing

1. What is Software Testing
2. Why Do we need testing
3. Basic Testing Terminologies
    1. Defect
    2. Vulnerability
    3. Requirement
    4. Design document
    5. Ambiguity analysis
    6. Review
4. The 7 Quality Principles and Concepts
5. The 2 software Quality Gaps
6. When Should Testing Occur
7. Testing Constraints
8. Validation and Verification process
9. Psychology of Testing

EgyTesters

What Is Software Testing

Testing has a lot of definitions as :

Testing software is operating the software under controlled conditions, to

- Verify that it behaves "as specified" ,
- To detect Defects / Errors ,
- To validate that what has been specified is what the user actually wanted.

EgyTesters

Why Do we need testing

Software is every ware controlling our as :

Work : PCs , OSs ,DBs , Servers , …
Social Life : FB , Twitter , Mobiles , **…**
Entertaining : Games , Play Pad , …
Medical : Medical tools , medical tracking systems , …
Other : Cars , Metro , Plains , trains , …

EgyTesters

Basic Testing Terminologies

1. Defect
2. Vulnerability
3. Requirement
4. Design document
5. Ambiguity analysis
6. Review

The 7 Quality Principles and Concepts

1. Testing shows presence of Defects
2. Exhaustive Testing is Impossible!
3. Early Testing
4. Defect Clustering
5. The Pesticide Paradox
6. Testing is Context Dependent
7. Absence of Errors Fallacy

The 7 Quality Principles and Concepts

1. Testing shows the presence of Defects

We test to find Faults (a.k.a Defects)

As we find more defects, the probability of undiscovered defects remaining in a system reduces.

However Testing cannot prove that there are no defects present

EgyTesters

The 7 Quality Principles and Concepts

2. Exhaustive Testing is Impossible!

We have learned that we cannot test everything (i.e. all combinations of inputs and pre-conditions).

That is we must Prioritise our testing effort using a Risk Based Approach.

EgyTesters

The 7 Quality Principles and Concepts

3. Early testing

Testing activities should start as early as possible in the development life cycle

These activities should be focused on defined objectives – outlined in the Test Strategy

Remember from our Definition of Testing, that Testing doesn't start once the code has been written!

EgyTesters

The 7 Quality Principles and Concepts

4. Defect Clustering

In most cases, it can be seen that the majority of the detected defects are caused by a small number of modules, i.e. the distribution of defects are not across the application but rather condensed in particular areas of the application.

This analogy is based on the Pareto principle, also known as the 80-20 rule, where it is stated that approximately 80 per cent of the problems are caused by 20 per cent of the modules

The 7 Quality Principles and Concepts

5. The Pesticide Paradox

Testing identifies bugs, and programmers respond to fix them

As bugs are eliminated by the programmers, the software improves

As software improves the effectiveness of previous tests erodes

Therefore we must learn, create and use new tests based on new techniques to catch new bugs

N.B It's called the "pesticide paradox" after the agricultural phenomenon, where bugs such as the boll weevil build up tolerance to pesticides, leaving you with the choice of ever-more powerful pesticides followed by ever-more powerful bugs or an altogether different approach.' – Beizer 1995

The 7 Quality Principles and Concepts

6. Testing is Context Dependent

Testing is done differently in different contexts

For example, safety-critical software is tested differently from an e-commerce site

Whilst, Testing can be 50% of development costs, in NASA's Apollo program it was 80% testing

3 to 10 failures per thousand lines of code (KLOC) typical for commercial software

1 to 3 failures per KLOC typical for industrial software

0.01 failures per KLOC for NASA Shuttle code!

Also different industries impose different testing standards

The 7 Quality Principles and Concepts

7. Absence of Errors Fallacy

If we build a system and, in doing so, find and fix defects ....

It doesn't make it a good system

Even after defects have been resolved it may still be unusable and/or does not fulfil the users' needs and expectations

The 2 software Quality Gaps

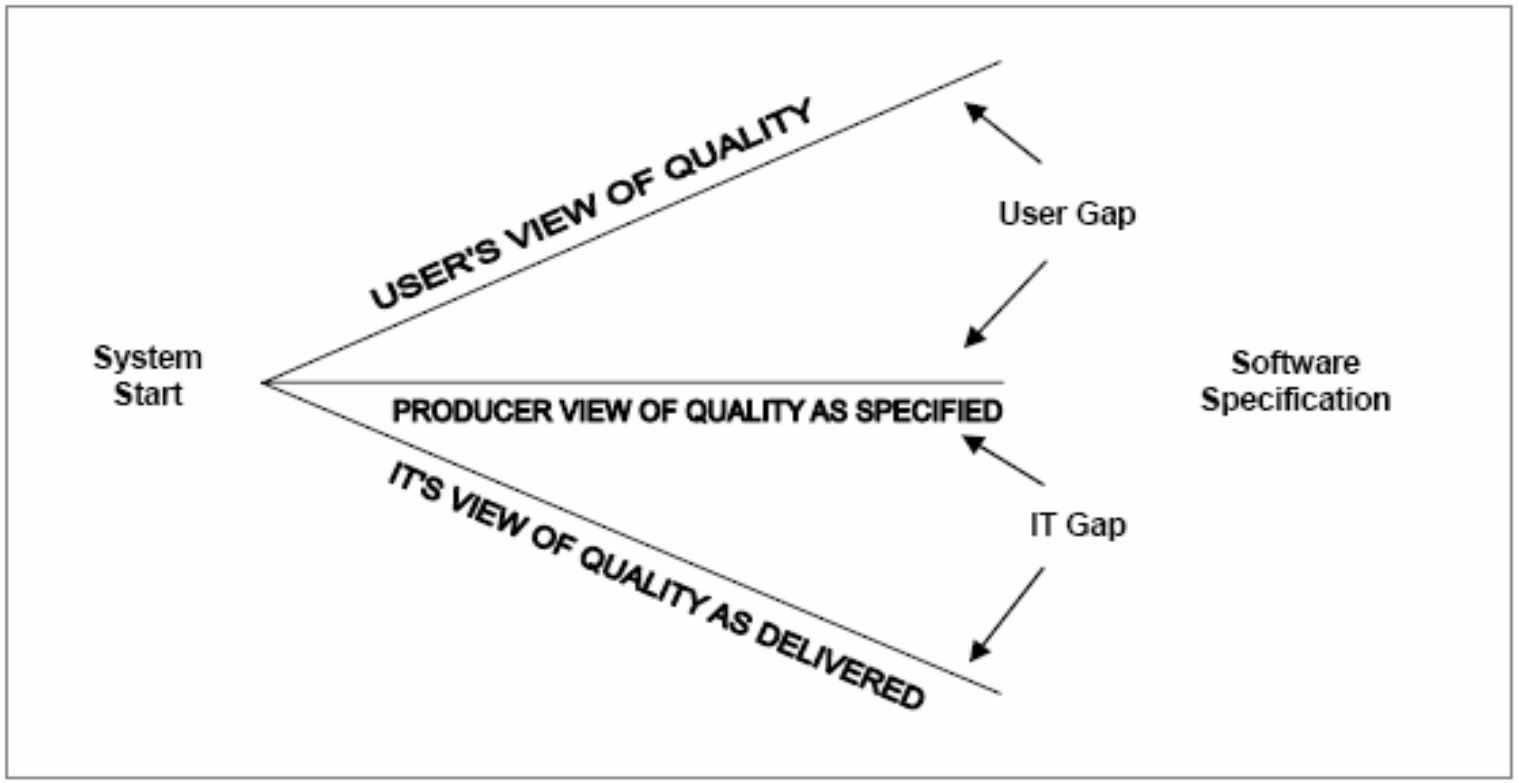


Figure 4. The Two Software Quality Gaps

EgyTesters

The 2 software Quality Gaps

- 1st gap is the IT gap. It is the gap between what is specified to be delivered, meaning the documented requirements and internal IT standards, and what is actually built.

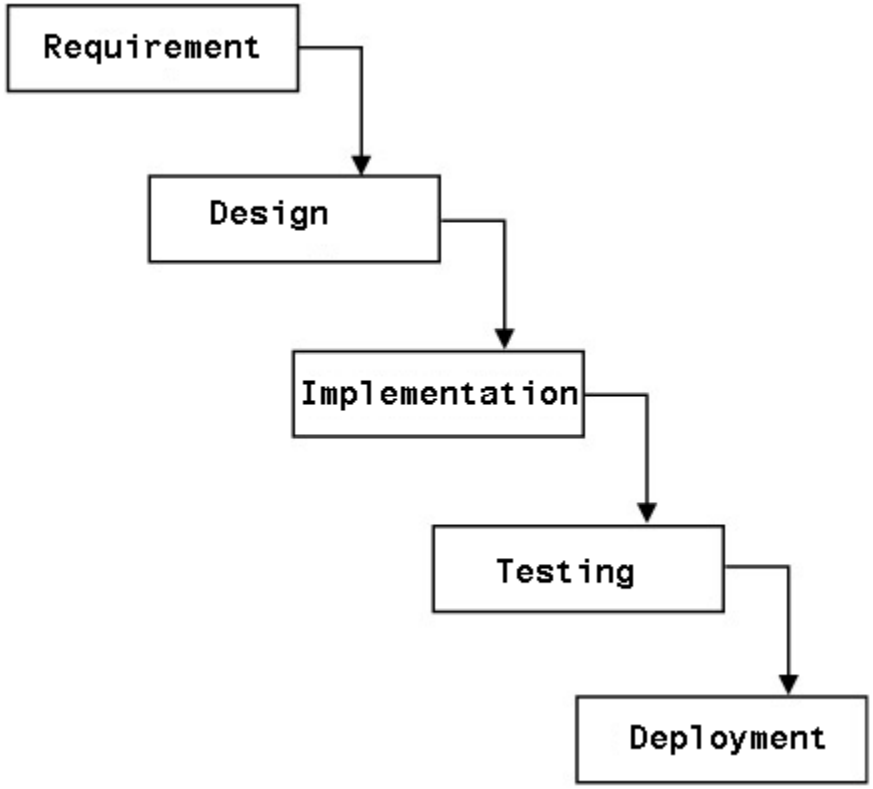
- 2nd gap is between what IT actually delivers compared to what the user wants.

EgyTesters

The 2 software Quality Gaps

- To close the user's gap we need :
  - Customer surveys
  - JAD (joint application development) sessions
  - Negotiate and agree upon requirements
  - More user involvement while building information products

When Should Testing Occur



EgyTesters

Testing Constraints

1. Limited Schedule & Budget
2. An incomplete statement of Work
3. Changes in technology
4. Limited tester Skills

EgyTesters

Validation and Verification process

- Verification
- The process of determining that the system does the right things
- Determining if the system complies with the requirements
- Most verification techniques are static tests.

- **Examples of static tests:**
  - Feasibility Reviews – Tests for this structural element would verify the logic flow of a unit of software.
  - Requirements Reviews – These reviews verify software relationships; for example, in any particular system, the structural limits of how much load (e.g., transactions or number of concurrent users) a system can handle.

Validation and Verification process

- **Validation**

- The process of determining that the system does things right
- To determine if the system is consistent, adheres to standards, and performs the selected functions in the correct manner.
- Most validation tests are dynamic tests.

- **Examples of Dynamic tests:**
  - System Testing – The tests simulate operation of the entire system and verify that it ran correctly.
  - User Acceptance – This real-world test means the most to your business, and

EgyTesters

Psychology of Testing

Why Testers Not Developers

The relationship between a Developer and a Tester is not normally an easy one because:

- Testers point out problems with software
- Developers like to think their software is perfect
- Testers are perceived as delaying the project by finding faults in the system

- It is important that they work together
- It is also important that they have mutual respect for each other.
- Collaboration is the right approach – we work to a common goal!
- Communicate findings objectively, not subjectively

EgyTesters

EgyTesters