QA Testing Boot Camp

Chapter 7 Fundamentals of Software Testing

Learning Outcomes

- Understand the reason for testing.
- Focus on the scientific principles behind the design/deployment of the testing initiative.
- Understand the variances between the different types of defects.

Why Testing is Necessary

Because it is developed by humans and humans make mistakes

All artifacts of the SDLC need to be examined for flaws

It supports the business on many levels

Testing Goals and Objectives

The major objectives are:

- Finding Defects
- Gaining Confidence
- Preventing Defects
- Meeting Objectives
- Satisfying the BRD/SRS
- Ensuring Testing is Cost Effective

What is a Defect (aka Bug)?

It is an error – intentional or unintentional

Source – many points of failure

- BRD/SRS
- Programming
- Project Management
- Change Control
- Etc.

Bottom-Line – actual result deviates from specification

What is a Defect? – The Defect Report

Lists all tests performed and the associated results.

Identifies all steps needed to recreate the issue. This is critically important.

> The ***programmer needs to be able to recreate the issue in order to determine how to apply a correction***.

GRAND CIRCUS
·DETROIT·

What is a Defect? – The Defect Report

Defect ID – Every defect has a unique identification number.

Defect Description – This field includes the abstract of the issue.

Product Version – This field includes the product version of the application in which the defect exists.

Detail Steps – This field includes the detailed steps of the issue with the screenshots attached so that developers can recreate it.

Date Raised – This field includes the date when the defect was reported.

Reported By – This field includes the contact information for the tester who reported the defect such as his name and ID number.

Status – This field includes the status of the defect. Possible entries are new, assigned, open, retest, verification, closed, failed, deferred, etc.

Fixed by – This field includes the contact information for the developer who corrected the issue.

Date Closed – This field includes the date when the defect was closed

Severity – This field identifies the impact of the issue on the software application. Possible entries include minor, major, and critical.

Priority – This field prioritizes the defect and influences how quickly the error will be addressed. High, medium, and low are typical entries in this field.

What is a Failure?

Not all defects result in failures

- Some lay dormant – **defects in dead code**

Failures can be caused by:

- Environmental conditions
- Hardware failure
- Software failure
- Bad data
- Intentional harm to the system

Difference: Defect vs. Error vs. Failure

Error – Programmer’s mistake

Defect – Bugs introduced in the code

Failure - Defect that when executed results in a false result or system crash

Differences in expected outcome may happen due to invalid data – this is not a failure – reported as an incident

Incident – condition where further review is required

How do we end up with Defects?

Arise from:

- Errors in the specification, design and implementation of the software and system
- Errors in use of the system
- Environmental conditions
- Intentional damage
- Potential consequences of earlier errors

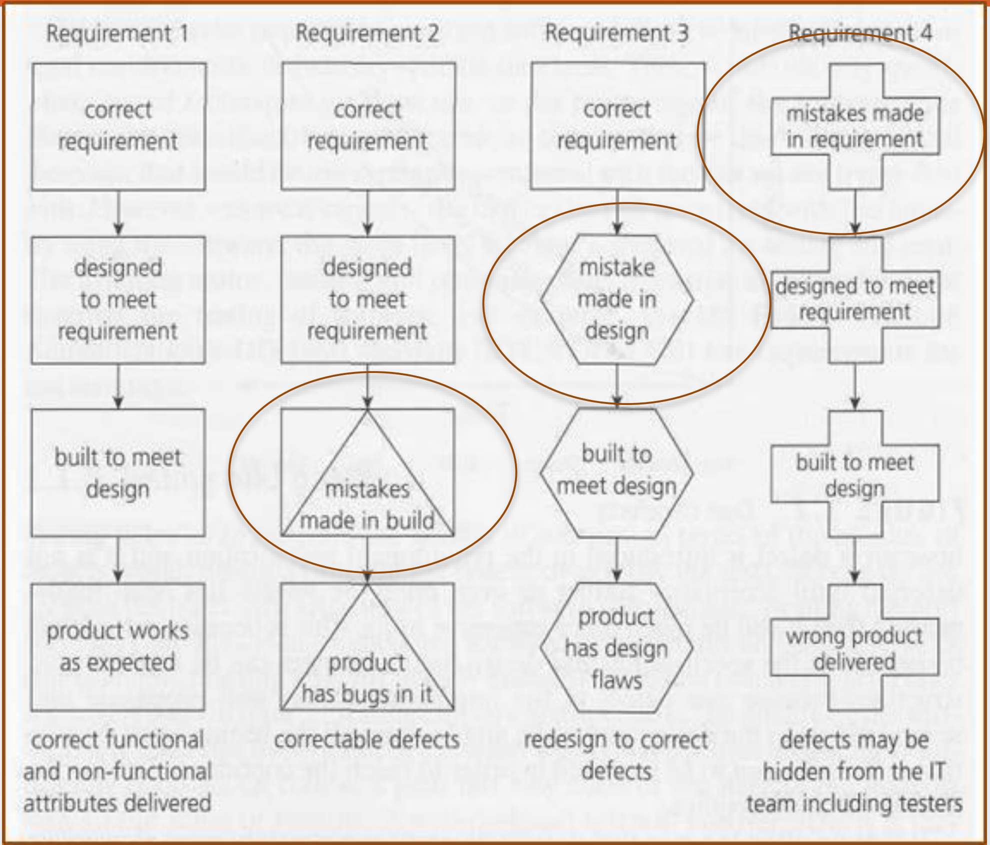
When Do Defects Happen?

Because:

- The person using the software application or product may not have enough knowledge of the product.
- Software is used in the wrong way which leads to the defects or failures.
- The developers may have coded incorrectly and there can be defects present in the design.
- Incorrect setup of the testing environments.

GRAND CIRCUS
·DETROIT·

When do Defects Happen – An Example



Requirement 1 – no defects

Requirement 2 – Fine until coding

Requirement 3 – Design defect

Requirement 4 – SRS defect

Costs Associated with Defect

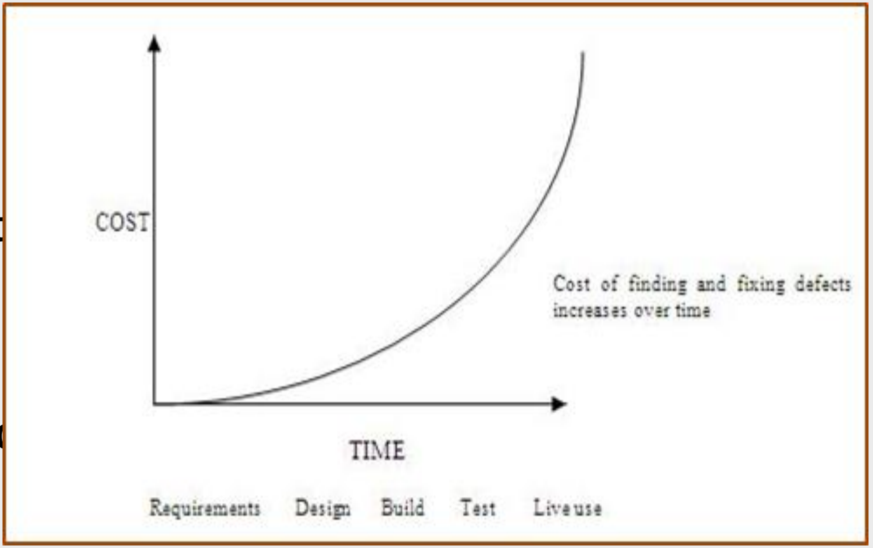
Measured by the impact of the defects and when we find them.

The earlier the defect is found lesser is the cost of defect.

Error is made/detected in the **requirements phase** then it is relatively cheap to fix it.

Error is made/detected in the **design phase or construction phase;** then the design can be corrected and reissued with relatively little expense.

Defect introduced in SRS/not detected until QA testing/implemented; much more expensive to fix.

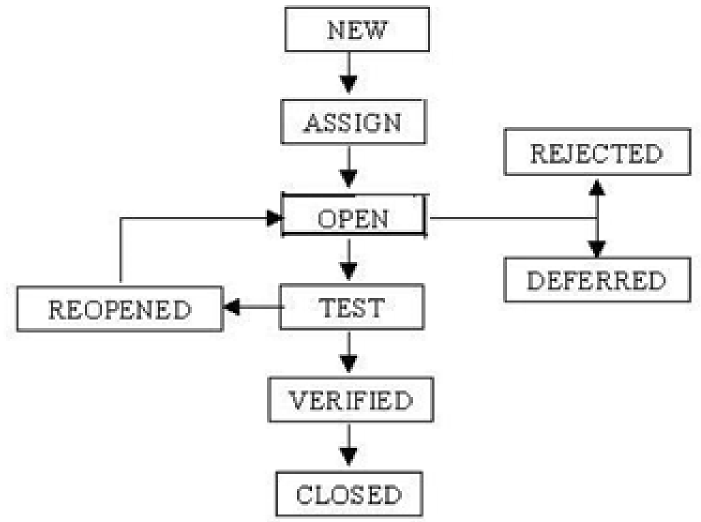


GRAND CIRCUS DETROIT

Defect Life Cycle

Starts when defect is found and ends when a defect is closed.

The phases of a defect are shown in the graphic.



Severity & Priority

Two key components of a defect:

Severity - **The extent to which the defect can affect the software – the impact.**

- **Critical**
- **Major**
- **Moderate**
- **Minor**
- **Cosmetic**

Priority - Defines the order in which we should resolve a defect – set by the tester.

- **Low**
- **Medium**
- **High**

Can have high severity but low priority.

Severity & Priority Possible Outcomes

High Priority & High Severity: An error which occurs on the basic functionality of the application and will not allow the user to use the system. (e.g. A student record that fails to save the record after editing is high priority and high severity bug.)

High Priority & Low Severity: Spelling mistakes that happens on the cover page or heading or title of an application.

High Severity & Low Priority: An error which occurs on the functionality of the application (for which there is no workaround) and will not allow the user to use the system. Occurs, however, on a path rarely executed by the end user.

Low Priority and Low Severity: Any cosmetic or spelling issues which are within a paragraph or a report (Not on cover page, heading, title).

Principles of Testing

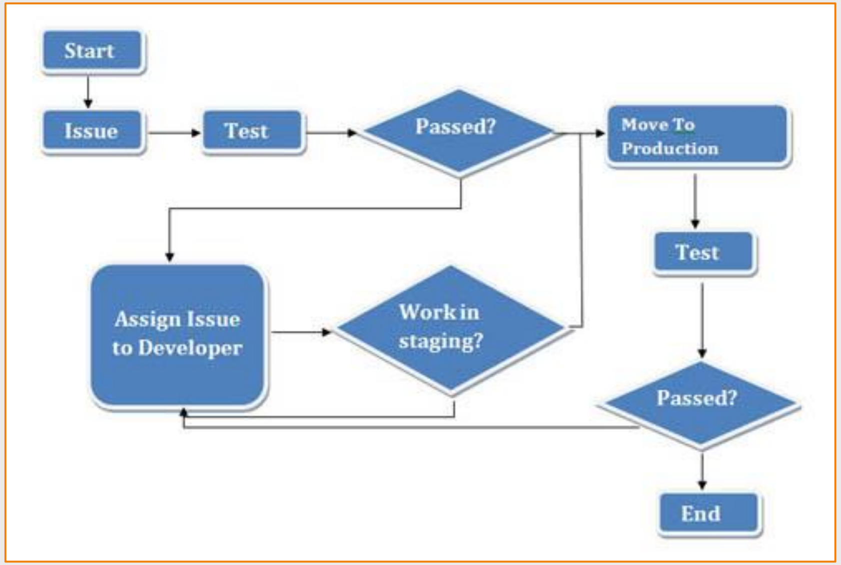
1) Testing shows presence of defects
2) EXHAUSTIVE TESTING IS IMPOSSIBLE!
3) Early testing
4) Defect clustering
5) Pesticide paradox
6) Testing is context depending
7) Absence of Errors Fallacy

Fundamentals of the Testing Process

Testing is often thought of as a single activity at the functional level as shown to the right.

It is really an overall principle of the entire SDLC that involves:

- Planning and Control
- Analysis and Design
- Implementation and Execution
- Evaluating Exit Criteria/Reporting
- Test Closure Activities



Fundamentals of the Testing Process

Planning and Control

- Determine scope, risks, objectives, test approach
- Build strategy
- Determine resources
- Schedule
- Determine exit criteria – coverage criteria

Fundamentals of the Testing Process

Analysis and Design

- Review the **test basis**
- Identify test conditions.
- Design the tests.
- Evaluate testability of the requirements and system.
- Design the test environment set-up and identify required infrastructure and tools.

Fundamentals of the Testing Process

Implementation and Execution

- Develop and prioritize test cases
- Create Test Suites
- Implement and verify the environment

Fundamentals of the Testing Process

Implementation and **Execution**

- Execute test suites and individual test cases
- Re-execute the tests that previously failed Log the outcome of the test execution
- Log the outcome
- Compare actual vs expected
- Report discrepancies as incidents

Fundamentals of the Testing Process

Evaluating exit criteria involves the following tasks:

- Check the test logs against the exit criteria specified in the test planning.
- Assess whether more test are needed, or if the exit criteria specified needs modification.
- Write a test summary report for stakeholders.

Fundamentals of the Testing Process

Test Closure Activities

- Verify that planned deliverables have indeed been delivered and ensure that all incident reports have been resolved.
- Finalize and archive test ware for later reuse.
- Handover the test ware to the maintenance organization. From this point forward, they will supply support for the software.
- Evaluate how the testing went and learn lessons for future releases and projects.

The Developer vs. The Tester

Testing and reviewing of applications are different from analyzing and developing.

Building the software requires a different mindset from testing the software.

This does not mean that the tester cannot be the programmer, or that the programmer cannot be the tester, though often different roles:

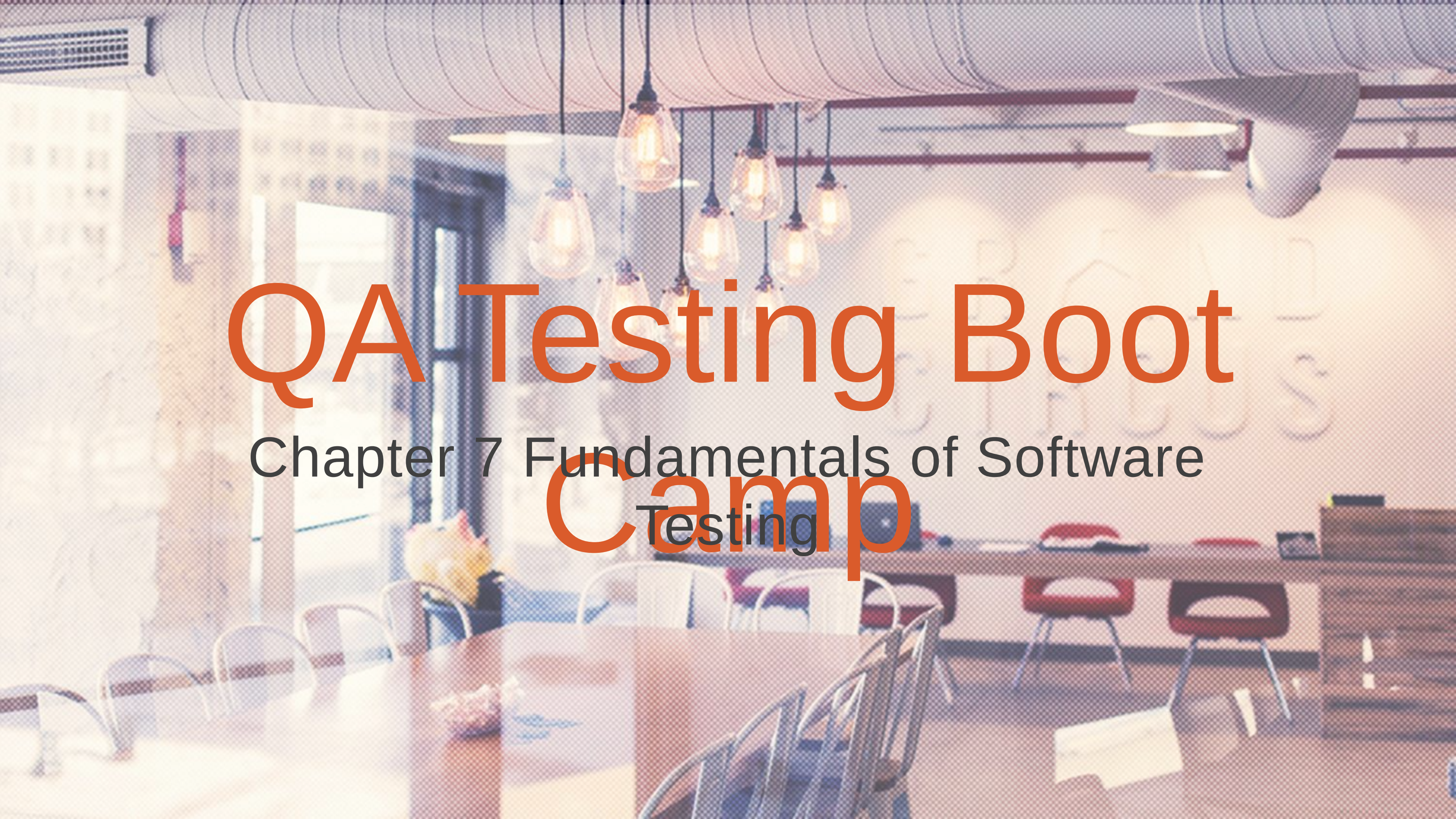
- Independence avoids author bias and often more effective
- communication between developer and tester is key.

Summary

In the preceding chapter, we learned what the foundational concepts of software testing include. Not only is it **finding defects in the software,** it is **finding defects in the design** as well as the requirements. We saw that not all defects result in failures and not all failures are defects.

Like all software processes, the defect goes through a lifecycle beginning with the discovery of the defect and continues through till the defect is considered to be closed.

We understand that programmers and testers, while often times are one in the same, are more effective when the roles are separated.



QA Testing Boot Camp

Chapter 7 Fundamentals of Software Testing