ISTQB – Foundation Level

CHAPTER 1: FUNDAMENTALS OF TESTING

Prepared by: Vu Nguyen
Updated by: Minh Ha
Dec 2012

AGENDA

ISTQB International Software Testing Qualifications Board

- 1.1 Introduction
- 1.2 Terms & Motivation
- 1.3 The Fundamental Test Process
- 1.4 The Psychology of Testing
- 1.5 General Principles of Testing
- 1.6 Testing Ethics

1.1 Introduction

ISTQB International Software Testing Qualifications Board

Why is testing necessary? Few examples:

- Mariner Bugs Out (1962):
  - Cost: $18.5 million
  - Disaster: The Mariner 1 rocket with a space probe headed for Venus diverted from its intended flight path shortly after launch. Mission Control destroyed the rocket 293 seconds after liftoff.
  - Cause: A programmer incorrectly transcribed a handwritten formula into computer code, missing a single superscript bar. Without the smoothing function indicated by the bar, the software treated normal variations of velocity as if they were serious, causing faulty corrections that sent the rocket off course.

1.1 Introduction

ISTQB International Software Testing Qualifications Board

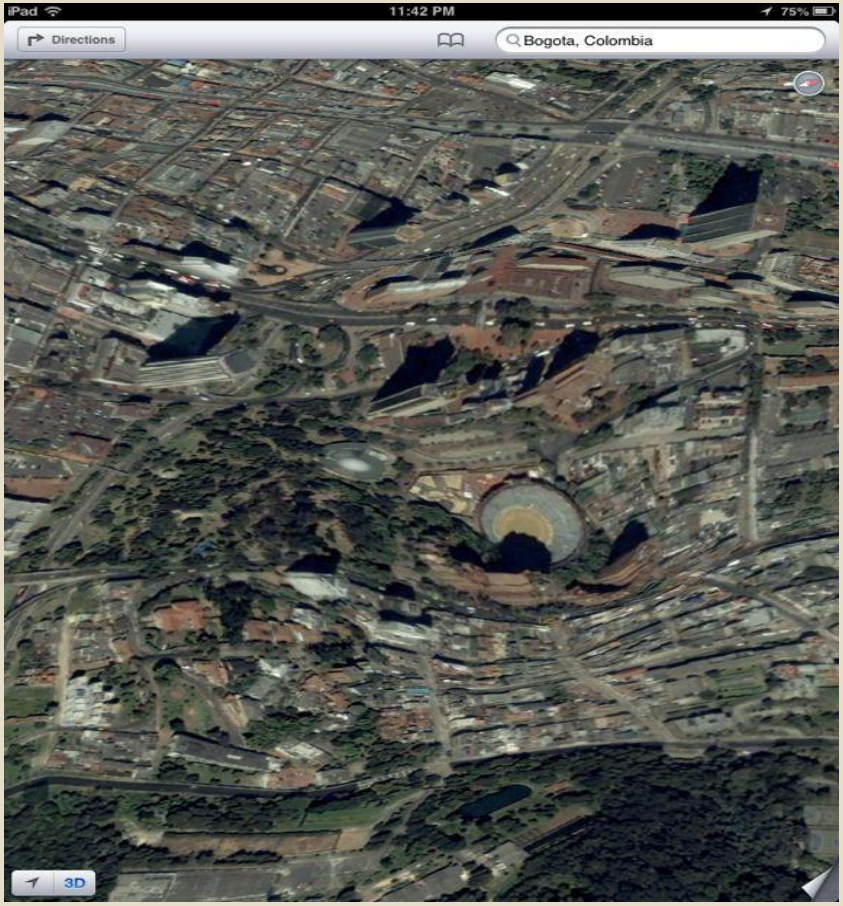
- Pentium Fails Long Division (1993)
  - Cost: $475 million, corporate credibility
  - Disaster: Intel Pentium chip made mistakes when dividing floating-point numbers within a specific range. Eventually Intel replaced the chips for anyone who complained.
  - Cause: The divider in the Pentium floating point unit had a flawed division table, missing about five of a thousand entries and resulting in these rounding errors.

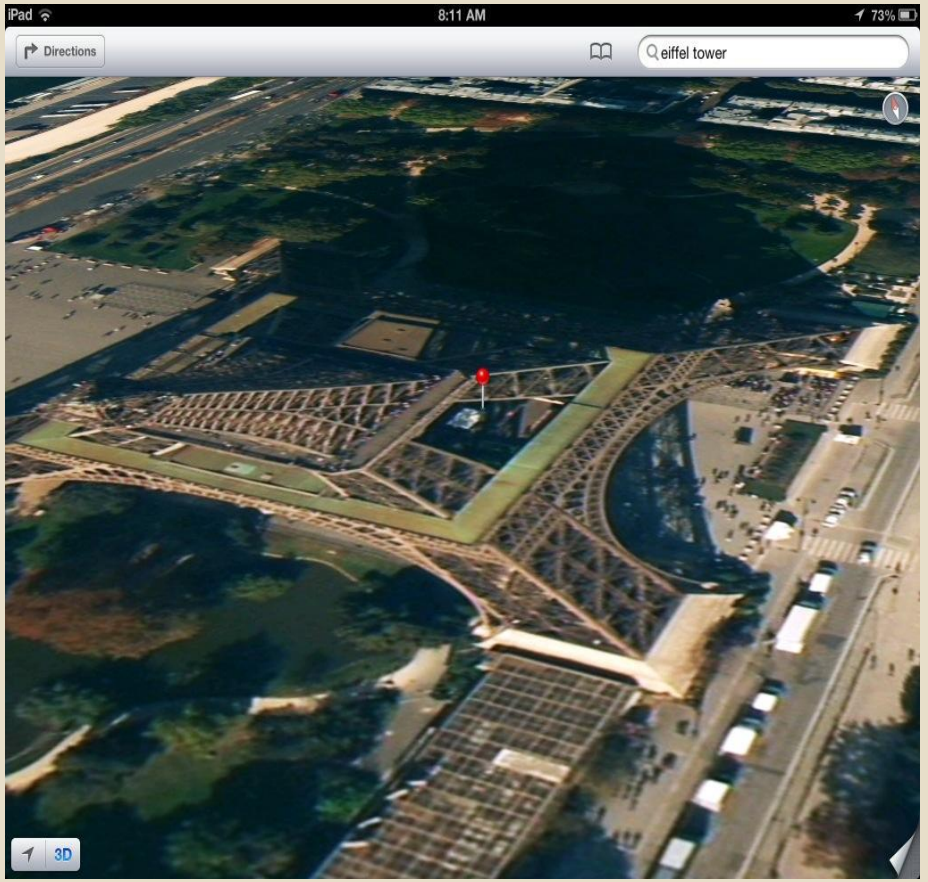
1.1 Introduction

- Y2K (1999)
  - Cost: $500 billion
  - Disaster: While no significant computer failures occurred, preparation for the Y2K bug had a significant cost and time impact on all industries that use computer technology.
  - Cause: To save computer storage space, legacy software often stored the year for dates as two digit numbers, such as “99" for 1999. The software also interpreted “00" to mean 1900 rather than 2000.

1.1 Introduction

ISTQB International Software Testing Qualifications Board





1.2 Terms and Motivation

- 1.2.1 Error and Bug Terminology
  - Failure:
    - A nonfulfillment of a given requirement; a discrepancy between the actual result/behavior and the expected result/behavior
    - A failure is present if a warrantable (user) expectation is not fulfilled adequately.
    - Other terms like “problem” or “incident” are often used

1.2 Terms and Motivation

- 1.2.1 Error and Bug Terminology
  - Fault:
    - We must differentiate between the occurrence of a failure and its cause.
    - A failure has its roots in a fault in the software.
    - Also called a defect or internal error or “bug”.
    - Caused by an error or mistake by a person or environmental conditions

1.2 Terms and Motivation

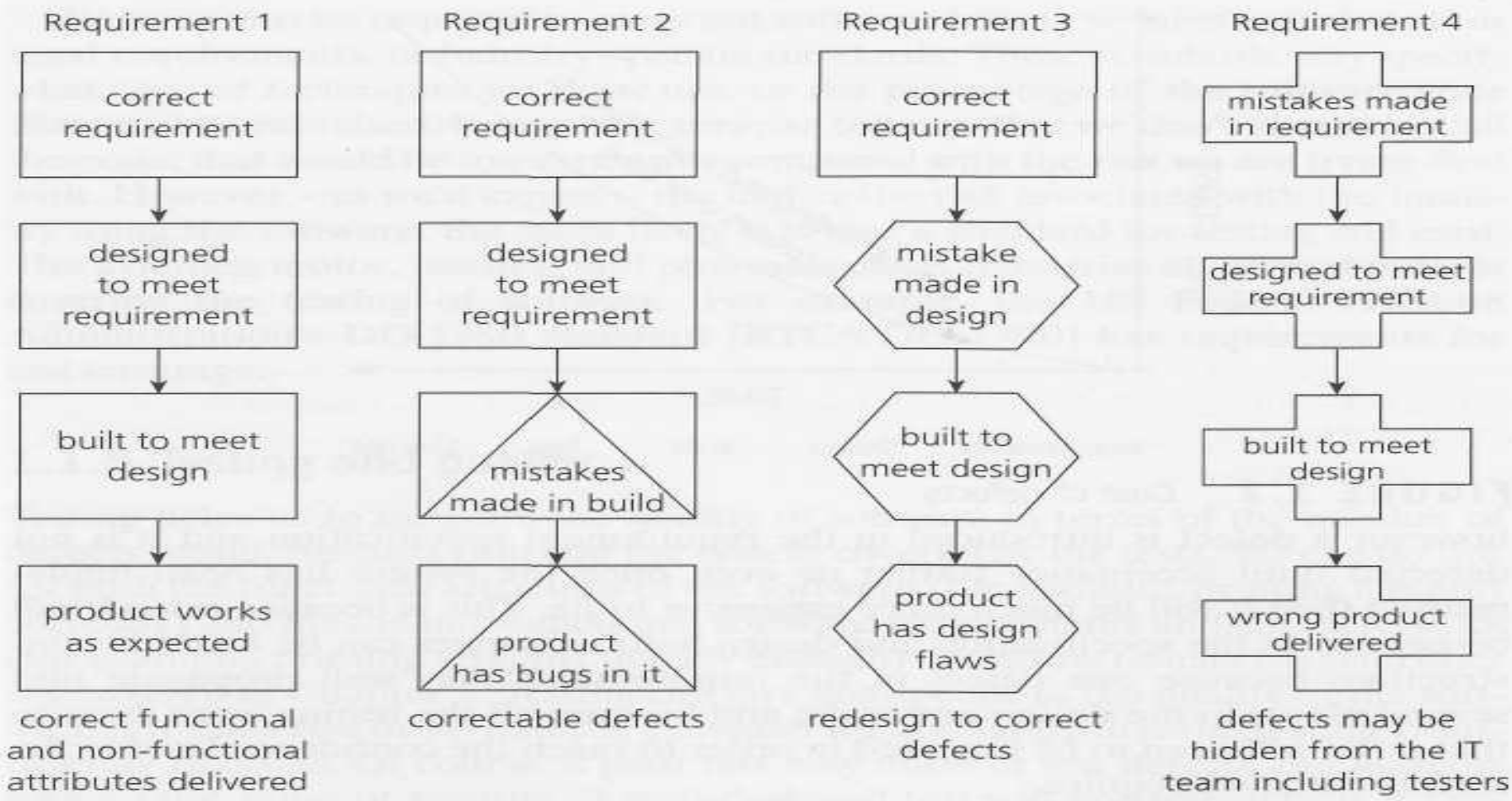
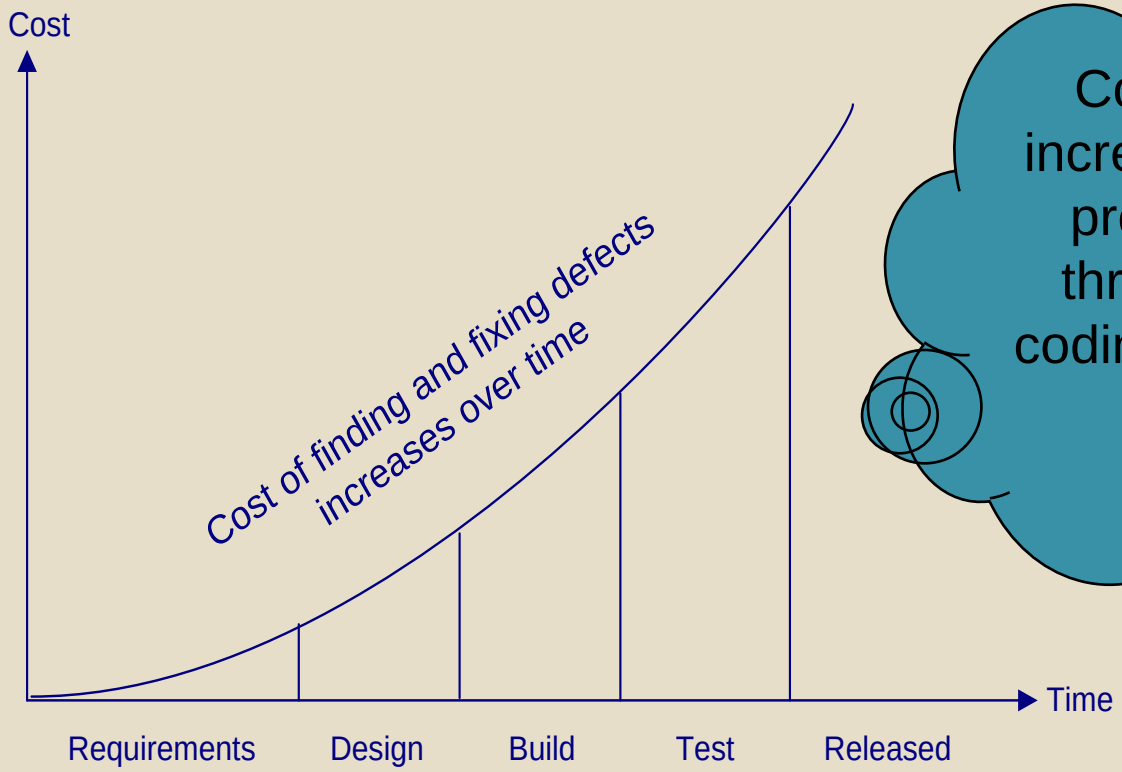


FIGURE 1.1 Types of error and defect

1.2 Terms and Motivation

- Cost of Defects



Cost becomes increasingly as the product moves through design, coding, testing, and to the field

1.2 Terms and Motivation

- 1.2.2 Testing Terms
  - Testing is not debugging
    - Debugging is the task of localizing and correcting faults.
    - The goal of testing is the detection of failures
    - Debugging of defects is the job of software developers.
    - Repairing a defect generally increases the quality of the product
    - Testing and debugging are totally different activities.

1.2 Terms and Motivation

- 1.2.2 Testing Terms
  - Testing software has different purposes:
    - Executing a program in order to find failures
    - Executing a program in order to measure quality
    - Executing a program in order to provide confidence
    - Analyzing a program or its documentation in order to prevent defects

1.2 Terms and Motivation

- 1.2.2 Testing Terms
  - Common terms:
    - Test process
    - Test case
    - Test suite or test run
    - Test condition, input, expect output/behavior, actual result
    - Test scenario

1.2 Terms and Motivation

- 1.2.2 Testing Terms
  - Bug free software:
    - No known bug free software system
    - Certain exceptional cases were not considered during development as well as during testing of the software
    - Even if all executed test cases do not reveal any further failures, cannot conclude with complete safety that there do not exist further faults

1.2 Terms and Motivation

- 1.2.3 Software Quality
  - Testing contributes to improvement of software quality
  - Testing is also measurement of software quality.
  - Following factors belong to software quality: functionality, reliability, usability, efficiency, maintainability, and portability.
  - All these factors have to be considered while testing.

1.2 Terms and Motivation

- 1.2.3 Software Quality
  - Functionality:
    - Contains all characteristics which describe the required capabilities of the system.
    - Described by a specific input/output behavior and/or an appropriate reaction to an input.
    - Interoperability describes the cooperation between the system to be tested and other systems.

1.2 Terms and Motivation

- 1.2.3 Software Quality
  - Security:
    - Fulfillment of application specific standards, agreements, or legal requirements and similar regulations.
    - Many applications give a high importance to the aspects of access security and data security.
    - Unauthorized access to applications and data, both accidentally and intentionally, will be prevented.

1.2 Terms and Motivation

- 1.2.3 Software Quality
  - Reliability:
    - Describes ability of system to keep functioning under specific use over a specific period.
    - Maturity
    - Fault tolerance
    - Recoverability.

1.2 Terms and Motivation

- 1.2.3 Software Quality
  - Maintainability
  - Portability
  - A software system cannot fulfill every quality characteristic equally well.
  - Quality characteristics must be prioritized

1.2 Terms and Motivation

- 1.2.4 Test Effort
  - Complete testing is not possible
  - Test effort between 25% and 50%
  - Faults can cause high costs
  - Define test intensity and test extent in dependence to the risk
  - Select adequate test procedures
  - Test of extra functionality
  - Limited resources

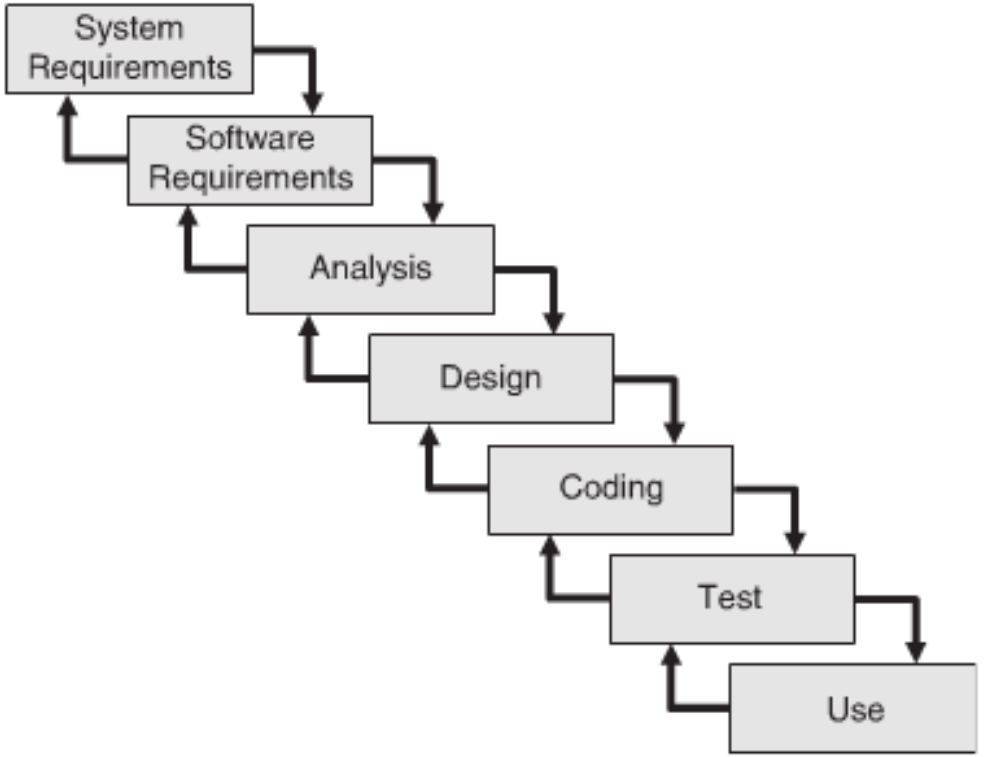
1.3 Fundamental Test Process

- Objectives
  - LO-1.4.1 Recall the fundamental test activities from planning to test closure activities and the main tasks of each test activity. (K1)

1.3 Fundamental Test Process

- There are many different software development models: Waterfall, V-model, Spiral, Agile…

Figure 2–3
Waterfall-model



1.3 Fundamental Test Process

- Fundamental test process is divided into the following basic steps:
  - Planning and control
  - Analysis and design
  - Implementation and execution
  - Evaluating exit criteria and reporting
  - Test closure activities
- Logically sequential, but may overlap, concurrently or be repeated

1.3 Fundamental Test Process

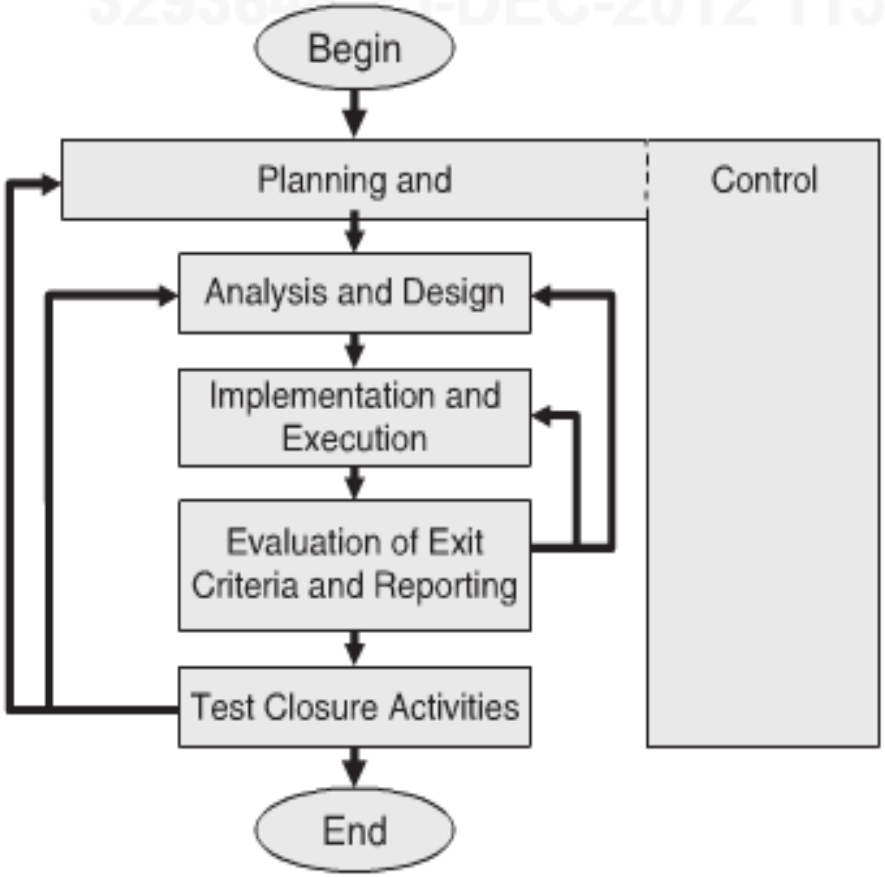


Figure 2–4
Fundamental test process

1.3 Fundamental Test Process

- Test planning
  - Determine the scope and risks and identify the objectives of testing
  - Determine the test approach
  - Implement the test policy and/or the test strategy
  - Determine the required test resources and tool support
  - Schedule test analysis and design tasks, test implementation, execution and evaluation
  - Determine the exit criteria

1.3 Fundamental Test Process

- Test control
  - Measure and analyze the results of reviews and testing
  - Monitor and document progress, test coverage & exit criteria
  - Provide information on testing
  - Initiate corrective actions
  - Make decisions

1.3 Fundamental Test Process

- Test analysis and design:
  - Review the test basis
  - Identify test conditions and design the tests
  - Test cases for expected and unexpected inputs
  - Evaluate testability of the requirements and system
  - Traceability is important
  - Design test environment set-up and identify any required infrastructure & tools

1.3 Fundamental Test Process

- Test implementation and execution
  - Implementation:
    - Develop and prioritize test cases
    - Create test suites
    - Implement and verify the environment

1.3 Fundamental Test Process

- Test implementation and execution
  - Execution:
    - Execute the test suites and individual test cases
    - The test execution must be exactly and completely logged
    - Compare actual results with expected results
    - The most important test cases first
    - Report incidents
    - Repeat test activities: confirmation testing or re-testing

1.3 Fundamental Test Process

- Evaluating exit criteria and reporting:
  - Check test logs against the exit criteria specified in test planning
  - Assess if more tests are needed
  - Write a test summary report

1.3 Fundamental Test Process

- Test closure activities:
  - Check planned deliverables and ensure all incident reports have been resolved
  - Finalize and archive testware
  - Hand over testware to the maintenance organization
  - Evaluate how the testing went and analyze lessons learned for future releases and projects

1.4 The Psychology of Testing

- People make mistakes, but they do not like to admit them!
- The tasks of developing software are often seen as constructive actions.
- The tasks of examining documents and software are seen as destructive actions.
- Developer test:
  - Who really likes to detect and show their own errors?
  - Blindness to one's own errors.

1.4 The psychology of testing

- Independent testing team:
  - An independent testing team tends to increase the quality and comprehensiveness of the tests.
  - Building the software requires a different mindset from testing the software
  - Independence avoids author bias and more effective at finding defects and failures
  - Several levels of independence

1.4 The psychology of testing

- Communicate findings on the product in a neutral, fact-focused way without criticizing the person who created it.
- Explain that by knowing about this now we can work round it or fix it so the delivered system is better for the customer.
- Start with collaboration rather than battles. Remind everyone of the common goal of better quality systems.

1.5 General Principles of Testing

- Principle 1: Testing shows the presence of defects, not their absence
  - Testing can show that the product fails, i.e., that there are defects. Testing cannot prove that a program is defect free.
  - Appropriate testing reduces the probability that hidden defects are present in the test object.
  - Even if no failures are found during testing, this is no proof that there are no defects.

1.5 General Principles of Testing

- Principle 2: Exhaustive testing is not possible
  - An exhaustive test where all values possible for all inputs and their combinations are run, combined with taking into account all different preconditions, is impossible.
  - The test effort must therefore be controlled, taking into account risk and priorities.

1.5 General Principles of Testing

- Principle 3: Testing activities should start as early as possible
  - Testing activities should start as early as possible in the software lifecycle and should focus on defined goals.
  - This contributes to finding defects early.

1.5 General Principles of Testing

- Principle 4: Defects tend to cluster together
  - Often, most defects are found in just a few parts of the test object.
  - Defects are not evenly distributed, but cluster together.
  - Many defects are detected in one place, there are normally more defects nearby.
  - During testing one must react flexibly to this principle.

1.5 General Principles of Testing

- Principle 5: The pesticide paradox
  - If the same tests are repeated over and over again, they tend to loose their effectiveness.
  - Previously undetected defects are not discovered.
  - New and modified test cases should be developed.
  - Parts of the software that until now were not tested, or previously unused input combinations will then be executed, and more defects may be found.

1.5 General Principles of Testing

- Principle 6: Test is context dependent
  - Testing must be adapted to the risks inherent in the use and environment of the application.
  - Therefore, no two systems should be tested in the exactly same way.
  - For every software system the test exit criteria, etc., should be decided upon individually, depending on its usage environment. Safety critical systems require different tests than e-commerce applications.

1.5 General Principles of Testing

- Principle 7: The fallacy of assuming that no failures means a useful system
  - Finding failures and repairing defects does not guarantee that the system as a whole meets user expectations and needs.
  - Early involvement of the users in the development process and the use of prototypes are preventive measures intended to avoid problems.

1.6 Tester Ethics

ISTQB International Software Testing Qualifications Board

- Involvement in software testing makes it possible for testers to get access to confidential and privileged information.
- A code of ethics is necessary, among other reasons to ensure that the information is not put to inappropriate use.
- Act in a manner that is in the best interest of client and employer, consistent with the public interest
- Ensure that the deliverables meet the highest professional standards possible.

1.6 Tester Ethics

- Maintain integrity and independence in professional judgment.
- Test managers and leaders shall subscribe to and promote an ethical approach to the management of software testing.
- Be fair to and supportive of their colleagues, and promote cooperation with software developers.

References

- Rex Black, Foundations of Software Testing
- ISTQB Foundation Syllabus.pdf

Q & A

Glossary

- **Test approach:** The implementation of the test strategy for a specific project. It typically includes the decisions made that follow based on the (test) project's goal and the risk assessment carried out, starting points regarding the test process, the test design techniques to be applied, exit criteria and test types to be performed.
- **Test basis**: All documents from which the requirements of a component or system can be inferred. The documentation on which the test cases are based. If a document can be amended only by way of formal amendment procedure, then the test basis is called a frozen test basis.

Glossary

- Test condition: An item or event of a component or system that could be verified by one or more test cases, e.g. a function, transaction, feature, quality attribute, or structural element.
- Test design specification: A document specifying the test conditions (coverage items) for a test item, the detailed test approach and identifying the associated high level test cases.
- Test design technique: Procedure used to derive and/or select test cases.

Glossary

- Test objective: A reason or purpose for designing and executing a test.
- Test policy: A high level document describing the principles, approach and major objectives of the organization regarding testing.
- Test strategy: A high-level description of the test levels to be performed and the testing within those levels for an organization or programme (one or more projects).

Glossary

- Test plan: A document describing the scope, approach, resources and schedule of intended test activities. It identifies amongst others test items, the features to be tested, the testing tasks, who will do each task, degree of tester independence, the test environment, the test design techniques and entry and exit criteria to be used, and the rationale for their choice, and any risks requiring contingency planning. It is a record of the test planning process.