Chapter 1 - ISTQB Fundamentals of testing

Lan Anh Tran

March 17, 2025

Contents

1 Introduction

Software is a big part of everyday life, but sometimes it doesn't work as expected. When software fails, it can cause problems like losing money, harming a company's reputation, or even putting people in danger. Testing helps find and fix these issues before the software is used. This chapter explains why testing is important, what causes software defects, and how testing can make software better. We will also look at how testing should be done at different stages and why independent testing is useful.

2 Why is testing necessary?

2.1 Software systems context

Definition 2.1. *A software system is a system of intercommunicating components based on software forming parts of a computer system.*

Software systems are an important part of life. Most people had experience with software not working as expected. If the software doesn´t work correctly, it can lead to problems like: loss of money, loss of business reputation, injury or death.

Testing´s contributions to success
The use of appropriate test techniques, applied with the right level of test expertise at the appropriate test levels in the software development life cycle, can be significant help in identifying problems so that they can be fixed before the software us released into use. Here are some examples where testing could contribute to more successful systems:

- Testers involved in requirements reviews or user story refinement could detect defects before any design or coding is done for the functionality described. Identifying and removing defects at this stage reduces the risk of the wrong software being developed.
- Testers working closely with system designers while the system is being designed can increase each party´s understanding of the design and how to test it. Since misunderstandings are often the cause of defects in software, having a better understanding at this stage can reduce risk of design defects.
- Testers working closely with developers while the code is under developmentcan increase each party´s understanding of the code and how to test it. As with design, this increases understanding.
- Verification and validation of the software can detect failures that might otherwise have been missed. This is traditionlayy where the focus of testing has been.

Quality assurance and testing
Is quality assurance the same as testing? No, the two are not the same.

Definition 2.2. *Quality assurance is a part of quality management focused on providing confidence that quality requirements will be fulfilled.*

2.2 Causes of software defects

Definition 2.3. *Error (mistake): A human action that produces incorrect results*

Definition 2.4. *Defect (bug, fault): An imperfection or deficiency in a work product where it does not meet its requirements or specifications*

Definition 2.5. *Failure: An event in which a component or system does not perform a required function within specified limits.*

Causes of software defects

Internal causes of software defects:

- Fatigue
- Lack of training
- Lack of understanding
- Lack of interest

External causes of software defects:

- Time pressurse
- Complex code
- Changed technologies
- Many systems interactions

Four types of scenarios

Figure 1 shows four typical scenarios, the upper stream being correct requirements, design and implementation, the lower three streams showing defect introduction at some phase in the software life cycle.

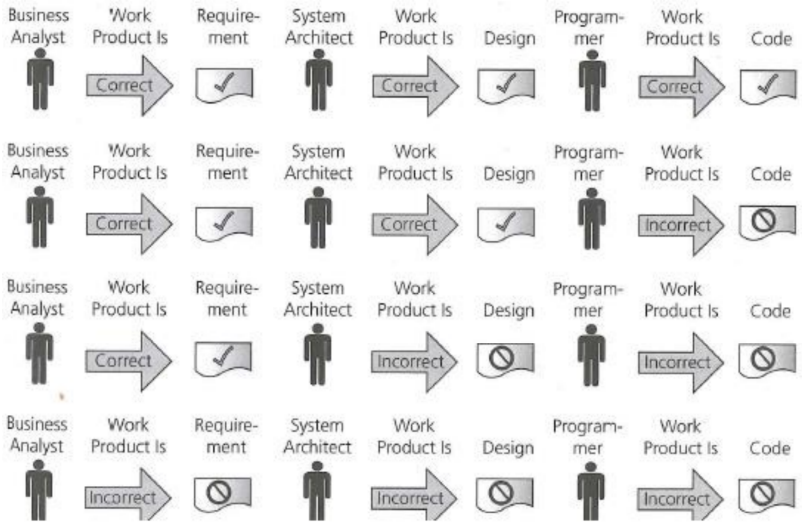

Figure 1: Four types of scenarios.

Cost to repair

Phase containment is important because the cost of finding and removing a defect increases each time that defect escapes to a later lyfe cycle phase. Multiplicative increases in cost, oft the sort seen in Figure 2, are not unusual.

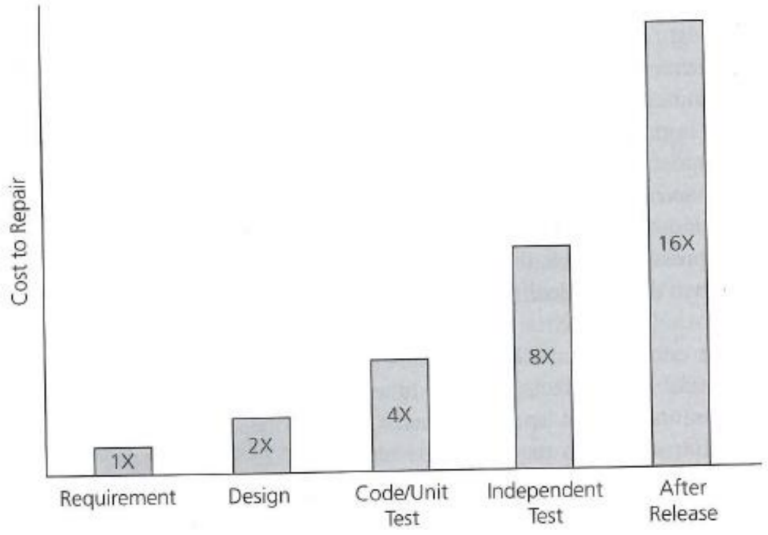

Figure 2: The cost to repair defects is lower at an early stage.

2.3 Role of testing in software systems

Testing has an important role in all stages of a software product´s life cycle:

- Planning
- Development
- Maintenance
- Operations

The role of testing is also to reduce the risk of problems occuring during operation. The ability to check if the software meets: legal requirements, industry specific standards. And off course to learn more about the software system.

2.4 Testing and quality

Testing measures the quality the software in terms of defects found.

- Functional aspects
- Non-functional aspects (reliability, usability, portability)

Testing creates confidence in the quality of the software

- If its properly tested and a minimum of defects are found

Testing teaches us lessons to apply in future projects

- By understanding the root causes of defects, processes can be improved. This can prevent defects from reoccurring.

2.5 How much testing is enough?

It is impossible to test everything! Testing should provide sufficient information for the stakeholders to make informed decisions about:

- Release of the software
- Next development steps, etc.

How much testing we should do depends on:
Level of risk:

- Technical risks
- Business risks

Project constraints:

- Time
- Budget

3 What is testing

3.1 Definition of testing

Definition 3.1. *The process of testing all software life cycle activites (both static and dynamic), concerned with: planning, preparation and evaluation of software products and related work products to determine that they satisfy specified requirements, to demonstrate that they are fit for purpose and to detect defects.*

Testing is more than running tests
An ongoing misperception, although less common these days, about testing is that it only involves running tests. But there are many other activites involved in the test process.

Testing is more than verification
Checking against a specification (called verification) is certainly part of testing, where we are asking the question, "Have we built the system correctly?". We also need to test to see if the delivered software and system will meet user and stakeholder needs and expectations in its operational environment. Often it is the tester who becomes the advocate for the end-user in this kind of testing, which is called validation. Here we are asking the question, "Have we built the right system?".

Test activites

- Test planning
- Test control
- Test analysis
- Test design

- Test implementation
- Test executing
- Checking results
- Evaluation of exit criteria
- Test results reporting
- Test closure

4 The seven test principles

The seven principles are in action on most if not all projects. Knowing how to spot these principles, and how to take advantage of them, will make you a better tester.

4.1 Principle 1: Testing shows the presence of defects, not their absence

Testing can show that defects are not present, but it cannot prove that there are no defects! Testing reduces the probability of undiscovered defects remaining in the software. However, even if no defects are found, this is not a proof of correctness.

4.2 Principle 2: Exhaustive testing is impossible

Testing everything is not feasible (capable of being done) except for secific cases. We use risks and priorities to focus the testing.

4.3 Principle 3: Early testing saves time and money

Testing activites should start as early as possible in the software or system development life cycle and should be focused on defined objectives.

4.4 Principle 4: Defects cluster together

A small number of modules contains most of the defects discovered during pre-release testing.

4.5 Principle 5: Be aware of the pesticide paradox

If the same test are repeated over and over again, the same test cases will no longer find any new bugs. To overcome this pesticide paradox, the test cases need to be regulary reviewed and revised, and new and different test need to be written to investigate different parts of the software.

4.6 Principle 6: Testing is context dependent

Testing is done differently in different contexts. For example, testing of safety-critical software is different from e-commerce site testing.

4.7 Principle 7: Absence-of-error fallacy

Finding and fixing defects does not help if the software system does not fulfills users needs and expectations.

5 Fundamental test processes

- Test planning
- Test monitoring and control
- Test analysis
- Test design
- Test implementation
- Test execution
- Test completion

5.1 Test planning, monitoring and control

A test plan encompasses questions like: what, how, when, by whom?
Who (roles and responsability): Testers, developers, projects managers, stakeholders, etc. are responsible for planning, executing and monitoring test activites.
What (scope and activites): The test plan includes what will be tested (scope) and how/when it will be tested. It also covers: test levels, test types, documentation and risk analysis.
Why (purpose and importance): To ensure quality, reliability and preformance of the software. To identify risks early. To allocate resources effectively and avoid delays.
When (timing and scheduling:) Test planning starts early in the software development lifecycle (SDLC). Testing is scheduled at different phases: unit testing (during development), system testing (after integration), user acceptance testing (before release)
Where (application and implementation:) In the project life cycle - integrated into DevOps, Agile or traditional waterfall models. Across different environments - Local dev machines, test servers, staging, production

5.2 Analysis and design

This process focuses on reviewing the test basis. What is necessary to test? This includes requirements, product architecture, products design, interfaces, risk analysis report, etc. Analysis: General test objectives are transformed into test conditions.

Example 5.1. *Authentication function: Test if input is correct, does the function include integer and string, etc.*

Design: Test cases, test environments (run test automatic or manual), get access to right test data, create tracebility, which test can we trace to the specification we use?

5.3 Implementation and execution

Implememnt:

- Group test into scripts (Definition 5.1)
- Prioritize the scripts
- Prepare test oracles (Definition 5.2)
- Write automated test scenarios

Definition 5.1. *Test Script is a set of pre-defined test steps, often written in a programming language, that automates the execution of test cases*

Definition 5.2. *Test oracle is a mechanism that helps determine whether a test has passed or failed by comparing actual results to expected results.*

Exeute:

- Run the tests and compare results with oracles
- Report incidents
- Repeat test activities for each corrected discrepancy
- Log the outcome of the test execution

5.4 Test completion

Evaluate:

- Assess test execution againts the defined objects
- Check if more testes are needed and/or exit criteria should be changed

Report:

- Its important to write or extract a test summary report for the stakeholders.

Test closure activites:

- To make test assets available for later use

6 The psychology of testing

6.1 What is a good tester?

Characteristics a good tester should have is curiosity, professional pessimism, good with attetions to details, good communication skills and experience at error guessing. A good tester should also be good with communication about defects and failure in a constructive way: fact-focused reports and review of findings

6.2 Independence in testing

A certian degree of independence is often more effective at finding defects and failures. However, the developer can very efficiently find bugs in their own code. The level of independence in testing depends on the objective of testing.

6.3 Independence test levels

We are usually bad at finding defects and failure on their own, especially on tests made self.

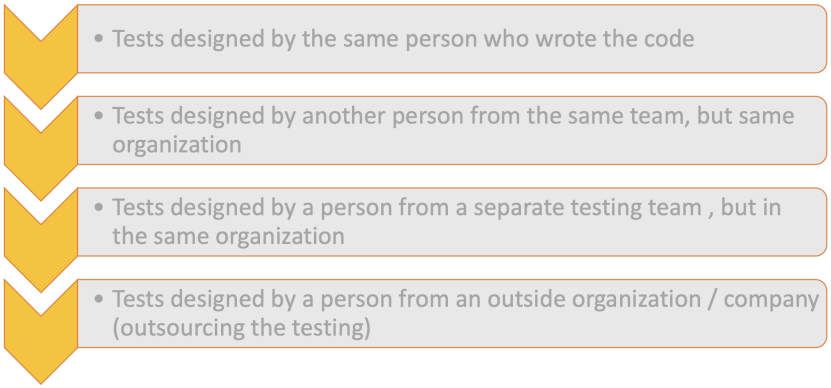

Figure 3: Test levels of independence

7 Conclusion

Testing is an important step in making good software. It helps find and fix mistakes early, saving time and money. In this chapter, we covered why testing is needed, the steps involved, and the principles that guide testing. We also talked about how testing improves software quality and reduces risks. In the end, good testing makes software more reliable and helps users feel confident in using it.