CHAPTER ONE

Fundamentals of testing

In this chapter, we will introduce you to the fundamentals of testing: what software testing is and why testing is needed, including its limitations, objectives and purpose; the principles behind testing; the process that testers follow, including activities, tasks and work products; and some of the psychological factors that testers must consider in their work. By reading this chapter you will gain an understanding of the fundamentals of testing and be able to describe those fundamentals.

Note that the learning objectives start with 'FL' rather than 'LO' to show that they are learning objectives for the Foundation Level qualification.

1.1 WHAT IS TESTING?

SYLLABUS LEARNING OBJECTIVES FOR 1.1 WHAT IS TESTING? (K2)

FL-1.1.1 Identify typical objectives of testing (K1)

FL-1.1.2 Differentiate testing from debugging (K2)

In this section, we will kick off the book by looking at what testing is, some misconceptions about testing, the typical objectives of testing and the difference between testing and debugging.

Within each section of this book, there are terms that are important – they are used in the section (and may be used elsewhere as well). They are listed in the Syllabus as keywords, which means that you need to know the definition of the term and it could appear in an exam question. We will give the definition of the relevant keyword terms in the margin of the text, and they can also be found in the Glossary (including the ISTQB online Glossary). We also show the keyword in **bold** within the section or subsection where it is defined and discussed.

In this section, the relevant keyword terms are **debugging**, **test object**, **test objective**, **testing**, **validation** and **verification**.

Software is everywhere

The last 100 years have seen an amazing human triumph of technology. Diseases that once killed and paralyzed are routinely treated or prevented – or even eradicated entirely, as with smallpox. Some children who stood amazed as they watched the first gasoline-powered automobile in their town are alive today, having seen people walk on the moon, an event that happened before a large percentage of today's workforce was even born.

Perhaps the most dramatic advances in technology have occurred in the arena of information technology. Software systems, in the sense that we know them, are a recent innovation, less than 70 years old, but have already transformed daily life around the world. Thomas Watson, the one-time head of IBM, famously predicted that only about five computers would be needed in the whole world. This vastly inaccurate prediction was based on the idea that information technology was useful only for business and government applications, such as banking, insurance and conducting a census. (The Hollerith punch-cards used by computers at the time Watson made his prediction were developed for the United States census.) Now, everyone who drives a car is using a machine not only designed with the help of computers, but which also contains more computing power than the computers used by NASA to get Apollo missions to and from the Moon. Mobile phones are now essentially handheld computers that get smarter with every new model. The Internet of Things (IoT) now gives us the ability to see who is at our door or turn on the lights when we are nowhere near our home.

However, in the software world, the technological triumph has not been perfect. Almost every living person has been touched by information technology, and most of us have dealt with the frustration and wasted time that occurs when software fails and exhibits unexpected behaviours. Some unfortunate individuals and companies have experienced financial loss or damage to their personal or business reputations as a result of defective software. A highly unlucky few have even been injured or killed by software failures, including by self-driving cars.

One way to help overcome such problems is software testing, when it is done well. Testing covers activities throughout the life cycle and can have a number of different objectives, as we will see in Section 1.1.1.

Testing is more than running tests

Testing The process consisting of all life cycle activities, 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.

An ongoing misperception, although less common these days, about **testing** is that it only involves running tests. Specifically, some people think that testing involves nothing beyond carrying out some sequence of actions on the system under test, submitting various inputs along the way and evaluating the observed results. Certainly, these activities are one element of testing – specifically, these activities make up the bulk of the test execution activities – but there are many other activities involved in the test process.

We will discuss the test process in more detail later in this chapter (in Section 1.4), but testing also includes (in addition to test execution): test planning, analyzing, designing and implementing tests, reporting test progress and results, and reporting defects. As you can see, there is a lot more to it than just running tests.

Notice that there are major test activities both before and after test execution. In addition, in the ISTQB definition of software testing, you will see that testing includes both static and dynamic testing. Static testing is any evaluation of the software or related work products (such as requirements specifications or user stories) that occurs without executing the software itself. Dynamic testing is an evaluation of that software or related work products that does involve executing the software. As such, the ISTQB definition of testing not only includes a number of pre-execution and post-execution activities that non-testers often do not consider 'testing', but also includes software quality activities (for example, requirements reviews and static analysis of code) that non-testers (and even sometimes testers) often do not consider 'testing' either.

The reason for this broad definition is that both dynamic testing (at whatever level) and static testing (of whatever type) often enable the achievement of similar project objectives. Dynamic testing and static testing also generate information that can help achieve an important process objective – that of understanding and improving the software development and testing processes. Dynamic testing and static testing are complementary activities, each able to generate information that the other cannot.

Testing is more than verification

Another common misconception about testing is that it is only about checking correctness; that is, that the system corresponds to its requirements, user stories or other specifications. Checking against a specification (called **verification**) is certainly part of testing, where we are asking the question, 'Have we built the system correctly?' Note the emphasis in the definition on 'specified requirements'.

But just conforming to a specification is not sufficient testing, as we will see in Section 1.3.7 (Absence-of-errors is a fallacy). 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?' Note the emphasis in the definition on 'intended use'.

In every development life cycle, a part of testing is focused on verification testing and a part is focused on validation testing. Verification is concerned with evaluating a work product, component or system to determine whether it meets the requirements set. In fact, verification focuses on the question, 'Is the deliverable built according to the specification?' Validation is concerned with evaluating a work product, component or system to determine whether it meets the user needs and requirements. Validation focuses on the question, 'Is the deliverable fit for purpose; for example, does it provide a solution to the problem?'

Verification
Confirmation by examination and through provision of objective evidence that specified requirements have been fulfilled.

Validation
Confirmation by examination and through provision of objective evidence that the requirements for a specific intended use or application have been fulfilled.

1.1.1 Typical objectives of testing

The following are some **test objectives** given in the Foundation Syllabus:

- To evaluate work products such as requirements, user stories, design and code by using static testing techniques, such as reviews.
- To verify whether all specified requirements have been fulfilled, for example, in the resulting system.
- To validate whether the test object is complete and works as the users and other stakeholders expect – for example, together with user or stakeholder groups.
- To build confidence in the level of quality of the test object, such as when those tests considered highest risk pass, and when the failures that are observed in the other tests are considered acceptable.
- To prevent defects, such as when early test activities (for example, requirements reviews or early test design) identify defects in requirements specifications that are removed before they cause defects in the design specifications and subsequently the code itself. Both reviews and test design serve as a verification and validation of these test basis documents that will reveal problems that otherwise would not surface until test execution, potentially much later in the project.

Test objective A reason or purpose for designing and executing a test.

- To find failures and defects; this is typically a prime focus for software testing.
- To provide sufficient information to stakeholders to allow them to make informed decisions, especially regarding the level of quality of the **test object** – for example, by the satisfaction of entry or exit criteria.
- To reduce the level of risk of inadequate software quality (e.g. previously undetected failures occurring in operation).
- To comply with contractual, legal or regulatory requirements or standards, and/or to verify the test object's compliance with such requirements or standards.

Test object The component or system to be tested.

These objectives are not universal. Different test viewpoints, test levels and test stakeholders can have different objectives. While many levels of testing, such as component, integration and system testing, focus on discovering as many failures as possible in order to find and remove defects, in acceptance testing the main objective is confirmation of correct system operation (at least under normal conditions), together with building confidence that the system meets its requirements. The context of the test object and the software development life cycle will also affect what test objectives are appropriate. Let's look at some examples to illustrate this.

When evaluating a software package that might be purchased or integrated into a larger software system, the main objective of testing might be the assessment of the quality of the software. Defects found may not be fixed, but rather might support a conclusion that the software be rejected.

During component testing, one objective at this level may be to achieve a given level of code coverage by the component tests – that is, to assess how much of the code has actually been exercised by a set of tests and to add additional tests to exercise parts of the code that have not yet been covered/tested. Another objective may be to find as many failures as possible so that the underlying defects are identified and fixed as early as possible.

During user acceptance testing, one objective may be to confirm that the system works as expected (validation) and satisfies requirements (verification). Another objective of testing here is to focus on providing stakeholders with an evaluation of the risk of releasing the system at a given time. Evaluating risk can be part of a mix of objectives, or it can be an objective of a separate level of testing, as when testing a safety-critical system, for example.

During maintenance testing, our objectives often include checking whether developers have introduced any regressions (new defects not present in the previous version) while making changes. Some forms of testing, such as operational testing, focus on assessing quality characteristics such as reliability, security, performance or availability.

1.1.2 Testing and debugging

Let's end this section by saying what testing is not, but is often thought to be. Testing is not **debugging**. While dynamic testing often locates failures which are caused by defects, and static testing often locates defects themselves, testing does not fix defects. It is during debugging, a development activity, that a member of the project team finds, analyzes and removes the defect, the underlying cause of the failure. After debugging, there is a further testing activity associated with the defect, which is called confirmation testing. This activity ensures that the fix does indeed resolve the failure.

Debugging The process of finding, analyzing and removing the causes of failures in software.