# Testing Policies

# for

# Traffic Light Optimization
# Imagine Interactive Systems (Iminsys)

**Prepared by Team Alpha:**

1. TK Lebelo **u15209190 (team leader)**
2. R Rammbuda **u16207948**

**Submitted to : Stacey Baror**
**Avinash Singh**
**Eunice Hammond**

**May - November 2020**

# Contents

# 1 Introduction

## 1.1 Purpose

The purpose of this document is to provide a clear statement as to Alpha requirements with regard to the testing process for software and to outline who is responsible for which parts of that process. The requirements stated in this document shall apply to all projects. In the specific requirement documents, however they may be adapted in line with the needs that apply to the particular project or product concerned.

Chapter 2 contains an overall summary of the testing process, based on the V-model. In that part of the document, we will indicate which test levels lie within the scope of responsibility of Alpha.

## 1.2 Why is testing required?

Testing forms an essential precondition to ensure the successful construction and implementation of information systems. The complexity of modern-day software is such that it is almost impossible to implement it correctly the first time around, without any form of verification.

## 1.3 What is testing?

Testing software takes the form of a process that is used to verify and validate that a software program, application or product:

- Fulfils the business and technical requirements set out in the contract documents, the requirements, the analysis and design documents
- Works as expected and can be used within its operational environment
- Has been implemented with the required non-functional software characteristics

In this regard, verification and validation must be interpreted in the sense of the ISO standard:

- Verification: Confirmation, through the provision of objective evidence, that the stated requirements have been fulfilled.

- Validation: Confirmation, through the provision of objective evidence, that the stated requirements for a specific, intended use or application have been fulfilled.

  We can interpret this by asserting that verification involves confirming that the software has actually been created, whilst validation sets out to establish that the correct software has been created.

## 1.4 Principles

- Testing to reveals defects
  Testing reduces the likelihood that the software contains undiscovered defects,but if no defects are found, this cannot be regarded as proof that no defects are present.

- Impossible to exhaustive testing
  Instead of carrying out extensive testing, risk analyses and priorities must be used in order to restrict the effort involved in carrying out tests to the tests that genuinely need to be carried out.

- Test on an early stage
  The testing activities must begin as early as possible within the software development cycle. This will ensure that the defects are detected at an early stage, with the result that rectifying the defects will be less costly.

- Categorising errors
  A small number of the modules contain the largest number of defects discovered during the prerelease tests and/or are responsible for the most operational errors.

- Testing is context-dependent
  The test method employed will depend on the context in which the product will ultimately be used.

- The absence-of-errors fallacy
  Tracing and rectifying defects will be of no benefit if the system is unusable and/or does not fulfil the needs and expectations of the end-users.

# 2 The testing process and overall responsibilities

## 2.1 Test driven

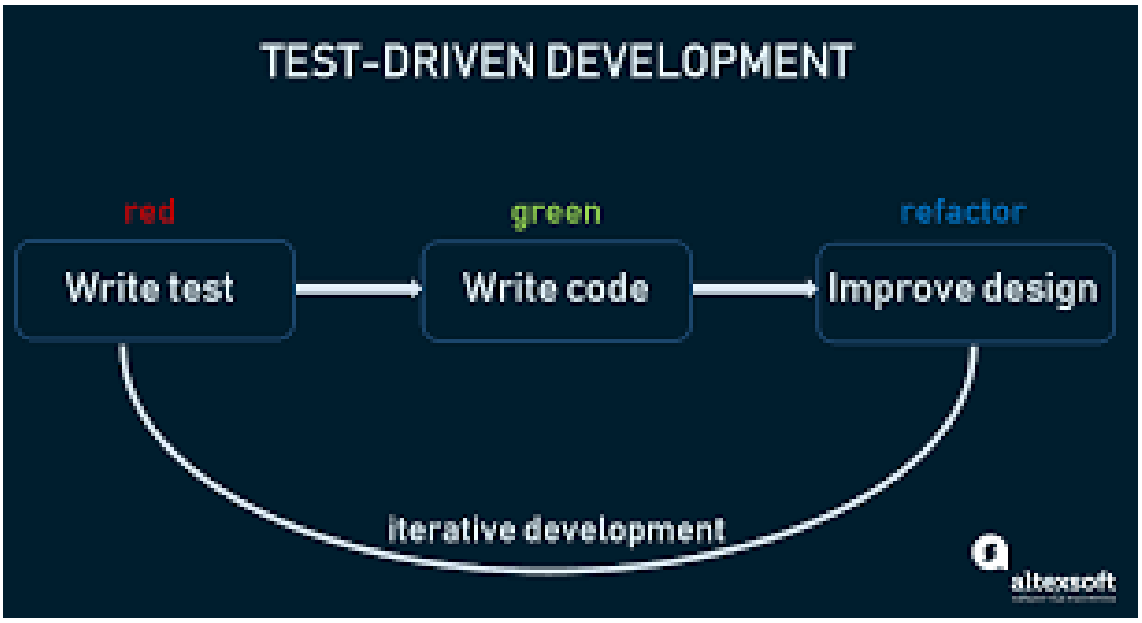


Figure 2.1: Process 1

## 2.2 Testing levels

Levels of testing take the form of groups of test activities that are collectively managed and directly related to the responsibilities defined as part of a project.

### 2.2.1 Reviews

Reviews form an excellent means of detecting faults at an early stage and in a cost-efficient way.

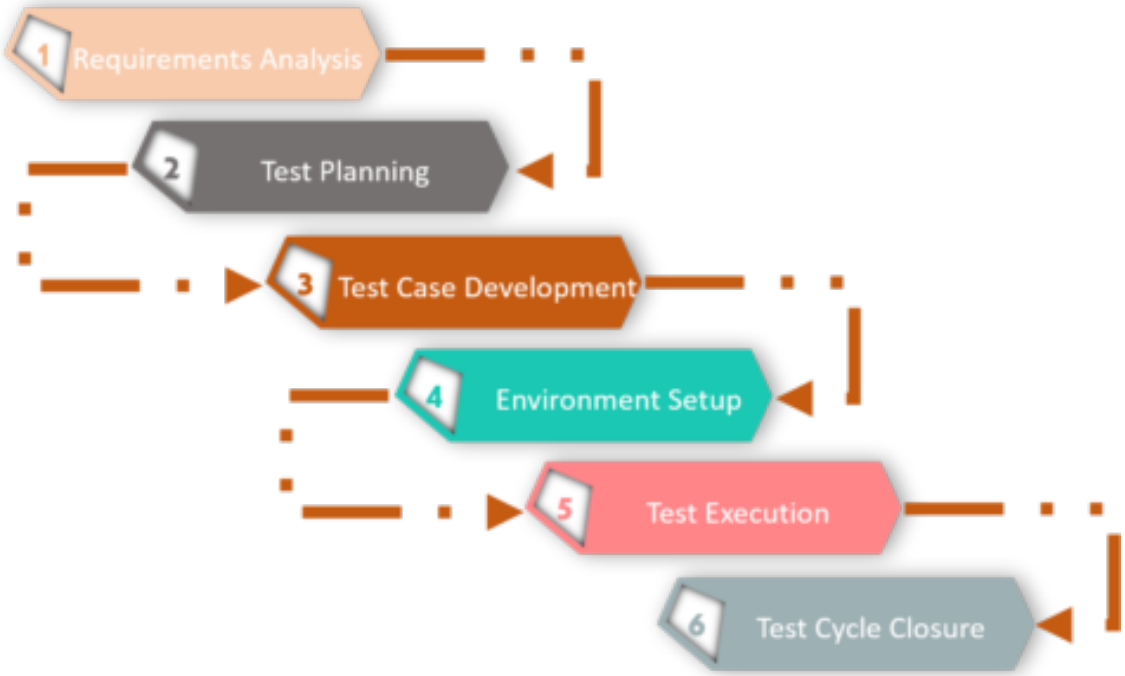


Figure 2.2: Analysis

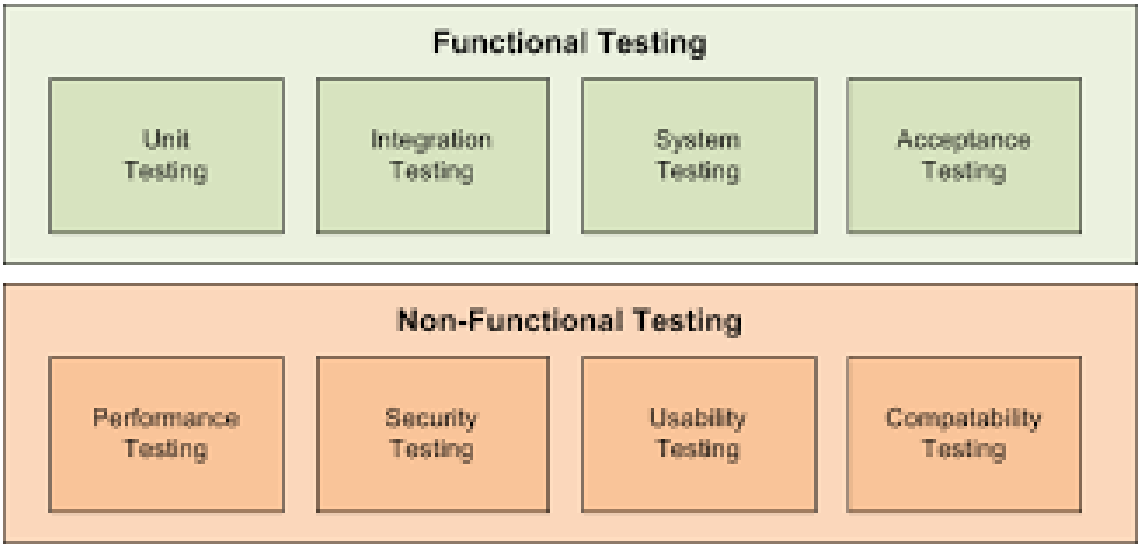


Figure 2.3: Process 3

### 2.2.2 Unit test level

Unit tests (otherwise known as component tests) seek to identify defects in, and verify the functioning of, testable software modules, programs, items, classes etc.
Unit tests are capable of verifying functionalities, as well as non-functional characteristics. The test cases are based on work products, such as the specifications of a component, the software design or the data model. Unit tests are used in order to be certain that the individual components are working correctly.

### 2.2.3 Integration test level

As far as the integration tests are concerned, these set out to test the interfaces between the various components and the interactions with different parts of a system (such as the control system, file system, hardware etc.). Integration tests can exist in more than one level and they can be carried out on test items of various sizes.

We are able to distinguish between 2 separate levels of integration test: - Component integration tests: The testing of the interaction between software components.These are carried out following the unit tests. - System integration tests: The testing of the interaction between various systems. These are carried out following the system tests.

The greater the scope of integration, the more difficult it becomes to isolate defects that exist in a specific component or system.

### 2.2.4 System test level

In the case of system tests, our aim is to test the behaviour of an entire system, as defined within the project. For the system tests, no knowledge is required, a priory of the internal design or logic of the system; the majority of techniques used are primarily black box techniques. In the case of the system tests, the environment must correspond as closely as possible with the ultimate production environment, in order to minimise the risk of environment-specific defects. System tests are derived from risk specifications, requirement specifications, business processes, use-cases or other high-level definitions and are carried out within the context of functional requirements. What is more, system tests will also investigate the non-functional requirements (quality attributes).
System tests typically include the following types of test: GUI tests, usability tests, regression tests, performance tests, error-handling tests, maintenance tests,compatibility tests, load tests, security tests, scalability tests, etc.

The intention is that once the system test has been carried out, the supplier is satisfied that the system complies with all of the necessary requirements and is ready to be handed on to the customer (possibly with the exception of interaction with peer systems).

## 2.3 Test Types

### 2.3.1 Functional tests

Functional tests set out to verify the functions that a system, sub-system or component is required to perform. Functional tests can be described in what are known as work products, such as requirement specifications, use-cases, functional specifications, etc.

These describe "what" the system actually does.

### 2.3.2 Non-functional tests

Non-functional tests encompass performance tests, load tests, stress tests, usability tests, maintainability tests, reliability tests (this list is not exhaustive).

Non-functional tests test "how" the system works.