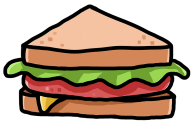


sandwico

Testing policy document: Real-time Fire Escape Route

by ERP

Mathilda Bresler
u16313382@tuks.co.za

Pieter Braak
u16313382@tuks.co.za

Kateryna Reva
u17035989@tuks.co.za

Jason Louw
u16313382@tuks.co.za

Xiao Jian Li
u16099860@tuks.co.za

20 May 2019

University Of Pretoria, Hatfield
Engineering, Built environment and Information Technology

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Contents

1 Introduction

1.1 Purpose of this document

Our team, Sandwico, believes that software testing is important, because a large amount of cost and resource can be saved - early discovery and correction of errors prevent spending a lot of time and money later during maintenance and repair. The policy defines the aims of testing, then describes the test process, mentions the test evaluation and the quality level to be achieved, and finally documents the approach to the improvement of the test process. The goal of testing is to adhere to the requirements, to create software that gives the users efficiency and user experience. The essence of the software testing approach is that testing activities start with the beginning of development life cycle, conforming to the requirement specification. It follows that testing tends towards compliance of requirements and defect reduction. The chosen test improvement process is TMMi. The testing process follows the chosen model and IEEE standards. The defined test levels are unit test, system test, and integration test. The above-mentioned three test levels are associated with each of the following activities:

- Planning, monitoring and control
- Analysis
- Design
- Implementation
- Execution
- Evaluating exit criteria and reporting
- Test closure activities

Regression testing is also used, mostly in unit testing. Reviews are also used on the specifications and other written work products. Test automation tools are implemented, which used generally to help execute unit and regression tests.

1.2 Intended Audience

Intended audience is:

- Team members – for testing, fixing bugs, implement changes
- The customers (lecturers and ERP Client) – for checking various functionalities
- The supervisor (Mentor) – to be informed about achievements and system functionality

1.3 Scope

This document contains information about how to ensure that our testing process is effective and that our software products meet the client's requirements we have developed.

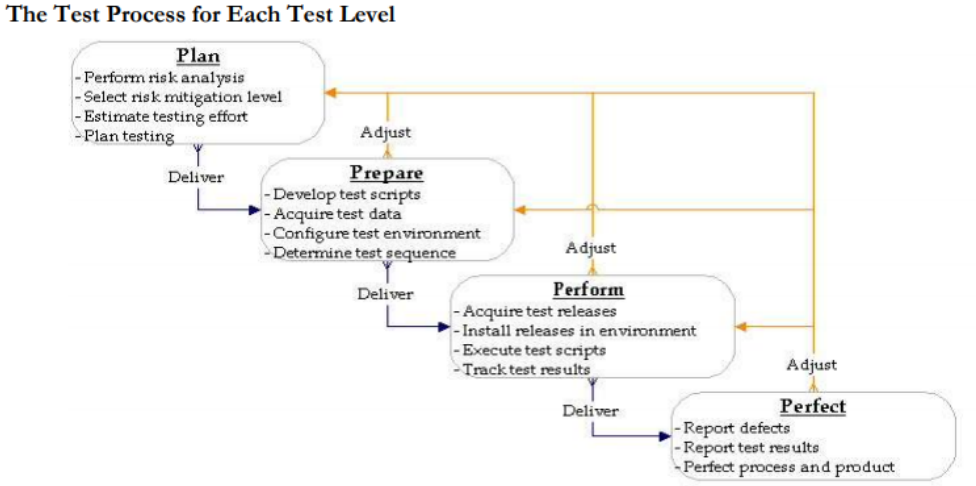
2 Test Policy statement/ goals

2.1 Testing approach

Testing should be done using Test Plan – all test data should be formed as described in particular case and preconditions should be fulfilled before testing. Results should be documented and as described in particular case.

Level	Owner	Objective(s)	Key Areas of Testing
Unit	Development	• Detect defective code in units • Reduce risk of unit failure in Production	Functionality and resource utilization
Integration	Development	• Detect defects in unit interfaces • Reduce risk of dataflow and workflow failures in Production	Functionality, data quality, unit interoperability and compatibility, performance
System	Risk Mitigation and Quality Assurance	• Detect defects in use cases and end-to-end scenarios • Assist in mitigating risk of unmet business requirements in Production Production	Functionality, data quality, performance, reliability, usability, resource utilization, maintainability, installability, portability and interoperability

2.1.1 Sequential Test Levels Performed by the Best-Qualified Participants



In our project the following apply regarding software testing:

- Random test should be perform After performing planned test cases, which should cover regular and irregular user actions and all features and exceptions, tester should try using the application. Testing procedure describe when and how to use random test.
- Random test of separate module before integration.
- Normal test – consists in testing the features trying to cover all the requirements.
- Faulty testing – consists in trying to feed the system with strange/invalid values, and see how the system react.
- Integration test – consists in testing modules after their integration in the system.
- Stress test – consists in simulating different access and watch if the system performance decrease or if the system crashes.

- Testing activities must be monitored using measurements and milestones to ensure that they are proceeding according to test plan.
- All features listed as requirements for all test items will be tested. Test cases are carefully chosen to cover all functional and nonfunctional requirements.

3 Testing software used

3.1 Travis CI

Travis CI is a hosted continuous integration service used to build and test software projects hosted at GitHub. Travis CI will monitor changes/pushes made to our Github repository and ensure that when those changes/pushes are committed that unit testing will automatically initialize.

Test being run on Travis CI

```
Note: Recompile with -Xlint:unchecked for details.

> Task :processResources NO-SOURCE
> Task :classes

> Task :compileTestJava
Note: /home/travis/build/cos301-2019-se/Real-time-Fire-Escape-Routes/src/test/java/HTTPServerTest.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

> Task :processTestResources NO-SOURCE
> Task :testClasses

> Task :test FAILED

DatabaseTest > DatabaseAddUser STARTED

DatabaseTest > DatabaseAddUser PASSED

DatabaseTest > DatabaseCreation STARTED

DatabaseTest > DatabaseCreation PASSED

DatabaseTest > DatabaseRemoveUser STARTED

DatabaseTest > DatabaseRemoveUser FAILED
    org.junit.ComparisonFailure at DatabaseTest.java:56

DatabaseTest > DatabaseReading STARTED

DatabaseTest > DatabaseReading PASSED

HTTPServerTest > testOne STARTED

HTTPServerTest > testOne PASSED

simpleTest > test1 STARTED

simpleTest > test1 PASSED

6 tests completed, 1 failed

FAILURE: Build failed with an exception.

* What went wrong:
Execution failed for task ':test'.
> There were failing tests. See the report at: file:///home/travis/build/cos301-2019-se/Real-time-Fire-Escape-Routes/build/reports/tests/test/index.html

* Try:
Run with --stacktrace option to get the stack trace. Run with --info or --debug option to get more log output. Run with --scan to get full insights.
```

3.2 Gradle

Gradle is an system builder which will identify all source files which will be used in order to automate the testing process on Travis CI

3.3 Junit Unit testing

JUnit is a simple framework to write repeatable tests. It is an instance of the xUnit architecture for unit testing frameworks.

The test were set up as the functional requirements emerged that would require functions.

The test were implemented as follows:

```java
public class ServerTester {
    private static Vector<Object> Classes;

    public static void main(String[] args) {
        JUnitCore.runClasses();
        Vector<Class> classesToTest = new Vector<>();
        classesToTest.add(WebAPITest.class);
        classesToTest.add(BuildingAPITest.class);
        classesToTest.add(BuildingGenerationAPITest.class);
        classesToTest.add(BuilderTest.class);

        System.out.println("================ Unit Tests ====================");

        for (int i = 0; i < classesToTest.size(); i++) {
            Class test = (Class) classesToTest.get(i);
            System.out.println("Currently testing: "+test.getName());
            Result Currentresult = JUnitCore.runClasses(test);
            if(Currentresult.wasSuccessful() == false) {
                for (Failure failure : Currentresult.getFailures()) {
                    System.out.println(failure.toString());
                }
            }
            System.out.println();
        }

        System.out.println("Testing HTTP Server: ");
        Result result = JUnitCore.runClasses(HTTPServerTest.class);

        for (Failure failure : result.getFailures()) {
```

The output received from the test display the functionality they are implemented to test.

```
Assign room to Building --  passed
Assign door to room --  passed
Getting distance in room --  passed
Getting number of people in room --  passed
Adding people to building --  passed
Getting num people in building --  passed
Getting number of people assigned to route --  passed
Getting door center --  passed
Getting person ID --  passed
Getting person position --  passed
Getting num people assigned to node --  passed
Getting distance to node --  passed
Getting node ID --  passed
true

Process finished with exit code 0
```

```
=============== Unit Tests ==================
Currently testing: test.WebAPITest
HandleRequest with valid JSON-- passed
HandleRequest with null jsonObject -- passed

Currently testing: test.BuildingAPITest
HandleRequest with valid JSON-- Pass
HandleRequest with null jsonObject -- Pass

Currently testing: test.BuildingGenerationAPITest
HandleRequest with valid JSON-- passed
HandleRequest with null jsonObject -- passed

Currently testing: test.BuilderTest
Builder/BuildPart Room -- passed
Builder/BuildPart Door -- passed
Builder/BuildPart Person -- passed

Currently testing: DatabaseTest
Databse file writing  --  passed
Databse file searching  --  passed
Databse creation  --  passed
Databse file reading  --  passed
DatabaseRemoveUser(DatabaseTest): expected:<[Json]> but was:<[Name Json not found
]>

Testing HTTP Server:
HTTPServer runs correctly --  passed
true
true

Process finished with exit code 0
```