CONTENTS

1 INTRODUCTION

1.1 Purpose of this Document

The purpose of this document is to provide an aid to the production of good quality test documentation by software engineering group projects.

1.2 Scope

This document specifies the standards for writing software test specifications and reports. It describes the necessary layout and content of these documents. It indicates the main stages of testing which should be carried out.

This document should be read by all project members. It is assumed that the reader is already familiar with the QA Plan [1].

1.3 Objectives

To describe the outline plan for testing, and the format of, and information which must be supplied in test specifications and test reports

2 RELEVANT QA DOCUMENTS

The Test Specification and Test Report must be produced in accordance with the quality standards specified in the QA Plan [1]. In particular, they must be maintained within the configuration management system according to the appropriate procedures [3]. The basic layout and information content must conform to the general documentation standards [4].

3 GENERAL APPROACH TO TESTING

Testing is used to establish the presence of defects in a program and it is also used to estimate whether or not a program is operationally usable. It is important to remember that testing can only demonstrate the presence of errors - it cannot prove their absence. Testing is thus used to “detect but not correct”. The function of testing is the discovery of errors, and the correction of errors should be left until *all* of the tests specified have been conducted. If errors are discovered, then the specified problem reporting and change control procedures must be followed in order to correct them [3]. Then, *all* the tests that exercise a system containing the amended items must be repeated. This is known as *regression testing*.

Particular emphasis should be placed on testing boundary situations, both inside and outside boundaries. For example, if a program is to process a maximum of 100 items typed in by the user, then test the program's behaviour for 0, 1, 100 and 101 items input as well as for some number of items between 1 and 100. Error conditions should also be tested - if it is possible for the user to type in illegal items, then tests should verify that such illegal input is detected and dealt with correctly.

Software on the group project will be subjected to three levels of testing: module, system, and acceptance. Module testing involves exercising a collection of related components (e.g. a package containing type definitions and subprograms), and the collection is tested in isolation from the rest of the system. In our context, a module is a Java class, and each class developed should have an accompanying Junit test class, which tests its functionality. Ideally, you should write this before designing and writing the code. This means that the programmer is responsible for module testing. Module tests do not need formal test plans or specifications.

System testing involves integrating all the modules together to form a complete system and then exercising that system.

It is desirable that system testing is carried out by persons other than those involved in the design or programming of the modules being exercised. This is true both for the production of test specifications, and for the execution of tests.

Acceptance testing involves exercising the entire system according to a set of procedures produced by the client, such that if all the tests are passed, the client agrees to accept the product. Acceptance testing is normally carried out by the client in the presence of all group members.

Sommerville provides some useful background information on testing in his textbook [5]. Pfleeger [6] also discusses both program testing and system testing in some detail.

4 TEST PLAN

Normally, members of a project would write a test plan outlining what testing needed to be done, and when. This is not necessary for the Group Project. For the group project, all project teams will follow this testing plan:

- Module testing will be left to the coder. In the case of Java code, should take the form of a set of JUnit tests that try all of the significant behaviour of the class. They should be written before the coding is done. For other languages, equivalent unit testing procedures should be defined.

- System testing will be done by writing a system Testing Specification during the design phase. This will cover all major functionality. When the system has been completed, a Test Report will document any features of the implementation that work incorrectly.

5 TEST SPECIFICATIONS

The purpose of a test specification is to specify in detail each of the system tests to be executed as part of a formal test process. The test specification must cross-reference to the appropriate section of the Requirements Specification for each feature being tested. The test specification provides a set of reproducible actions to test all of the main functionality of the system.

Each test specification will have an introductory section followed by a collection of test procedures. The introductory section will have the form as specified in QA Document SE.QA.03 [4], and must include a list of the documents from which the Test Specification is derived. Full bibliographic details of the documents can be provided in the reference list at the end of the document, but each document must be referenced in the introductory text.

Each individual test must be described in detail. This description is known as a *test procedure*. It specifies in detail how the test will be carried out.

Test Ref	Req being tested	Test Content	Input	Output	Pass Criteria
SE-F-001	FR1	Check that system can store the first two days of the earliest permitted year	Enter 1st March 1971 at date prompt. Hit return, and enter 2nd March 1971 at date prompt.	List of stored dates should now include those dates.	Data is stored correctly
SE-F-002	FR1	Check that system can store the last two days of the latest permitted year	Enter 30th December 2072 at date prompt. Hit return, and enter 31st December 2072 at date prompt.	List of stored dates should now include those dates.	Data is stored correctly
SE-F-003	FR2	Dates too early are rejected	Enter 1st January 1971.	Error message warns of date too early. List of stored dates should not have been changed.	System displays "wrong date" error message.

etc.					

6 TEST RESULT REPORTING

Details of test results must be maintained in a Test Report folder. This must have two sections labelled *Module Tests*, and *System Tests* respectively. Test reports should be stored electronically in the version control repository.

On completion of coding, the tests should be run, and the results submitted with the final report as a Test Report (see [2]).

7 BASELINES

It is most important that all items comprising a system to be tested are frozen into an identified (tagged) configuration. You can use the `git tag` command for this. The documents cross-referenced in the test specification (e.g. the Requirements Specification) also
form part of the configuration. This is described in QA Document SE.QA.08 [3], and whilst tests are being conducted, the tester must ensure that the have the correct version checked out. System level tests should be conducted periodically, particularly as the system is nearing completion. A test log form should be competed (Appendix A) each time system tests are conducted and stored in the appropriate folder in the version controlled repository. An issue should be raised on github for any tests that fail.

Appendix A - EXAMPLE TEST LOG FORM

TEST LOG FORM

Test Log No: 001	Group: 07	Testers(s): bpt, cjp
Date: 25/09/13	Tagged version ID: 0.1	

Test ID	Pass / Fail	Fail description	CCF / issue #
SE-F-001	Pass	N/A	N/A
SE-F-002	Pass	N/A	N/A
SE-F-003	Fail	Date too early is accepted.	#191

Test log form available on Group Project Blackboard Website

REFERENCES

[1] QA Document SE.QA.01 - Quality Assurance Plan.

[2] QA Document SE.QA.11 - Producing a Final Report.

[3] QA Document SE.QA.08 - Operating Procedures and Configuration Management Standards.

[4] QA Document SE.QA.03 - General Documentation Standards.

[5] I. Sommerville, Software Engineering, Addison-Wesley, 7th edition, 2004.

[6] S. L. Pfleeger, Software Engineering: Theory and Practice,Prentice Hallk, 2nd edition, 2001.

DOCUMENT HISTORY

Version	*CCF No.*	*Date*	*Changes made to document*	*Changed by*
1.0	N/A	09/10/01	Document given complete overhaul in Word	CJP
1.1	N/A	18/07/02	Changed after review with Graham Parker - minor corrections - removed subsystem testing and some verbosity - simplified testing plan	CJP
1.2	N/A	11/08/03	Test plan deleted and JUnit added.	CJP
1.3	N/A	26/09/04	References updated. BN12 added para p5.	CJP
1.4	N/A	14/09/06	Updated ref [2] to match new numbering	CJP
1.5	N/A	12/09/08	Changed document template to be Aber Uni	CJP
1.6	N/A	21/09/10	Converted to Docbook. SVN implication for baselines c.f. configurations.	NWH
1.7	N/A	24/09/13	Reverted to word doc. Updated unit testing to Junit, not Main method in class. Altered Test Log Form to a single table for all system level tests.	BPT
1.8	N/A	23/09/14	Unit tests broadened beyond Java.	NWH