**Software Test Plan (STP) Template**

Items that are intended to stay in as part of your document are in **bold**; explanatory comments are in *italic* text. Plain text is used where you might insert wording about your project.

This document is an annotated outline for a Software Test Plan, adapted from the IEEE Standard for Software Test Documentation (Std 829-1998).

Tailor as appropriate. Where you decide to omit a section, you might keep the header, but insert a comment saying why you omit the element.

**Ford-Paredes Group**

****

**freecol-0.11.6**

**Software Quality Assurance Plan**

**Version: (0.11.6) Date: (5/7/2016)**

**Document History and Distribution**

1. Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision # | Revision Date | **Description of Change** | **Author** |
| 1 | 5/7/2016 | Task1 completed – Repository for the collaboration platform opened. |  |
| 2 | 5/7/2016 | Task2 completed – Project application was selected and source code acquired. |  |
| 3 | 5/8/2016 | Task3 completed – The code was put into Eclipse, and preliminary coverage testing and metrics made to determine which class/methods are the most used. |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

1. Introduction 1

2. Test Items 2

3. Features To Be Tested 3

4. Features Not To Be Tested 3

5. Approach 3

6. Pass / Fail Criteria 5

7. Testing Process 5

8. Environmental Requirements 6

9. Change Management Procedures 7

# Introduction

(Note 1: The Software Test Plan guidelines were derived and developed from IEEE Standard for Software Test Documentation (829-1998)).

*(Note 2: The ordering of Software Test Plan (STP) elements is not meant to imply that the sections or subsections must be developed or presented in that order. The order of presentation is intended for ease of use, not as a guide to preparing the various elements of the Software Test Plan. If some or all of the content of a section is in another document, then a reference to that material may be listed in place of the corresponding content.)*

*This document will be providing the Software Test Plan for an open source application called FreeCol. This application is a turn-based strategy game based on the old game named Colonization, which is similar to another game called Civilization. The objective of the game is to create an independent nation. A player start with only a few colonist, defying stormy seas in their search for new land. The player(s) will guide them on the colonialization of the new world.*

*The source code for this application will be acquired and the quality assurance process performed on it. The Javadoc documentation will be examined to understand how the application is structured. A static analysis and a dynamic analysis will be performed as well to determine further how the quality assurance process must proceed.*

**The Software Test Plan (STP) is designed to prescribe the scope, approach, resources, and schedule of all testing activities. The plan must identify the items to be tested, the features to be tested, the types of testing to be performed, the personnel responsible for testing, the resources and schedule required to complete testing, and the risks associated with the plan**.

**1.1 Objectives**

*The application will undergo a static analysis as well as a dynamic analysis in an effort to make improvements on the application. After which testing will be performed to ensure faults are detected and removed.*

**1.2 Testing Strategy**

**Testing is the process of analyzing a software item to detect the differences between existing and required conditions and to evaluate the features of the software item.**

Since the application is already developed, the testing should consists of static analysis at the start to determine the direction of the testing. Once it’s determined that the dynamic part is ready to proceed, we proceed.

**1.3 Scope**

*(Specify the plans for producing both scheduled and unscheduled updates to the Software Test Plan (change management). Methods for distribution of updates shall be specified along with version control and configuration management requirements must be defined.)*

**Testing will be performed at several points in the life cycle as the product is constructed. Testing is a very 'dependent' activity. As a result, test planning is a continuing activity performed throughout the system development life cycle. Test plans must be developed for each level of product testing.**

**1.5 Definitions and Acronyms**

*Definitions of all packages and classes are given in the Javadoc’s provided by the following links; http://www.freecol.org/javadoc/*

# Test Items

*The test items will be the classes that’s constantly being used. (Specify the test items included in the plan. Supply references to the following item documentation:*

* *Users guide http://www.freecol.org/documentation/freecol-user-manual.html*
* *Javadoc documentation http://www.freecol.org/javadoc/*

**2.1 Program Modules**

*No modules will be tested.*

**2.2 User Procedures**

*The user’s guide for FreeCol is provided from the following link: http://www.freecol.org/documentation/*

# 3. Features To Be Tested

*The components that will be tested will consists of those classes that are constantly used by the application. Their methods will be tested as well.*

# 4. Features Not To Be Tested

*All other components not satisfied above will not be tested.*

# 5. Approach

*To start off the cyclomatic complexities of the different classes in this application need to be determined, to see if methods need to be refactored. Classes that yielded complexity of at least 7 will usually be flagged, and in turn we examined those classes closely. The methods that caused for the class to be flagged will be scrutinize further, and if refactoring was necessary should be done. This can be accomplished using CodePro tools installed on our development environment. From this we would obtain a document summarizing our result under the file name “****MetricResult****” which is in html format. Additional static analysis runs will be performed using CodePro to generate additional artifacts for review, e.g.* ***CodeProAuditResults****,* ***RefactoringsAuditResults****, and* ***SecurityAuditResults****. After this static analysis is performed, we refactor the necessary methods and unit test them to ensure that the units produces what we expect them to produce. To ensure that the units are working properly unit testing will be performed on the refactored methods. After a satisfactory coverage is assessed, a regression testing will be performed. This will be followed by the acceptance testing.*

**5.1 Component Testing**

*Unit testing will be performed using JUnit installed in Eclipse.*

**5.2 Integration Testing**

*Integration testing will not performed.*

**5.3 Interface Testing**

*Interface testing will only be performed at a less intensive level.*

**5.4 Security Testing**

*Security testing will be performed for the application from the static analysis portion of the testing activities. Auditing tools from CodePro will be used to perform this assessment.*

**5.5 Performance Testing**

*Some of the metrics measure here might not be critical since the game that this application is based upon is in all general purposes static they might not be essential.*

**5.6 Regression Testing**

*When refactoring the methods are completed and tested, regression testing will then commencement.*

**5.7 Acceptance Testing**

*Acceptance testing will be kept at a minimum. Running the application to play the game.*

**5.8 Beta Testing**

*No beta testing will be performed for the application.*

# 6. Pass / Fail Criteria

*If a coverage of at least 80% is obtained the application will deem as passing the testing process.*

**6.1 Suspension Criteria**

Not necessary for this application.

**6.2 Resumption Criteria**

Not necessary for this application.

**6.3 Approval Criteria**

*Approval will be given if the passing criteria above is obtained.*

# 7. Testing Process

*(Identify the methods and criteria used in performing test activities. Define the specific methods and procedures for each type of test. Define the detailed criteria for evaluating test results.)*

**7.1 Test Deliverables**

*Static Analysis Artifacts:*

1. *MetricResults*
2. *CodeProAuditResults*
3. *RefactoringsAuditResults*
4. *SecurityAuditResults*

*Dynamic Analysis Artifacts:*

1. *Final coverage report*

**7.2 Testing Tasks**

*After a static analysis is made to the code, if refactoring is needed unit testing is performed to ensure that the new units implemented interact appropriately with the other units. This is will be followed by a regression testing. And finally, acceptance testing will be conducted. Thus completing the testing activities.*

**7.3 Responsibilities**

*All responsibilities would fall on this group.*

**7.4 Resources**

*The resources used here includes the source code for the application and the development environment Eclipse*

**7.5 Schedule**

*Testing activities will not be schedule-driven, but event-driven. After static analysis is made unit testing will be conducted. After the required coverage is attained, acceptance testing is performed.*

# 8. Environmental Requirements

Testing activities will be conducted in our development environment Eclipse.

**8.1 Hardware**

*There are no computer hardware or network requirements needed to complete testing activities.*

**8.2 Software**

*The source code for the application will be embedded in Eclipse.*

**8.3 Security**

*There are no security requirements to identify on the testing environment.*

**8.4 Tools**

*The tools used in the testing environment are those used in a development environment, such as Eclipse along with plugins used in both qualitative and quantitative analysis.*

**8.5 Risks and Assumptions**

*There are no risk assumptions.*

# 9. Change Management Procedures

*Not necessary for this application.*