Group DuffyGaleEncinas

FreeCol

Software Quality Assurance Plan

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

Version: (1.2) **Date: (05/16/2016)**

Document History and Distribution

1. Revision History

Revision #	Revision Date	Description of Change	Author
1.0	05/15/2016	Initial Draft	Marlene Encinas
1.1	05/16/2016	Update over all document	Matthew Duffy
1.2	05/16/2016	Update over all document	Matthew Gale

TABLE OF CONTENTS

1. INTRODUCTION

Software testing is an essential activity to produce high-quality software. Using automated tools assist to achieve this goal by eliminating drudgery, saving time, eliminating error by omission, and eliminating some of the discrepancies in test quality caused by differences in individual's abilities.

The document describes the procedure and expectations for testing FreeCol version 0.11.6, open source, strategy game based on the old game "Colonization", and similar to "Civilization".

FreeCol mostly uses Java as programming language and it is platform independent. It is known to run on Linux and Windows, as well as Mac OS X (with some limitations).

This Test Plan uses the IEEE standard for Software Test documentation (829 -1998) to delineate a plan to follow.

1.1 Objectives

The major testing objective is to use static code analyzer tools to improve the existing code, make it more readable, understandable and clean.

The scope of the project focuses on reasonable number of modules that offer complexity level 7 or higher and refactor the modules. Based on the time and schedule provided to perform this testing only a limited number of classes where modified.

1.2 Testing Strategy

Considering the constraints described in section 8.5, the purpose of this test is based on refactoring FreeCol code.

The testing strategy will be a combination of using automated test tools and using built-unit test from FreeCol.

Google Code Pro will be the central analytic tool to analyze dependencies, compute metrics and calculate the Cyclomatic Complexity (CC). The work activity will be identifying methods with CC of 7 or higher and bringing them down less than 7. Other tools such as FindBugs will also be used to find potential bugs in the program.

1.3 Scope

The scope of this document is to test FreeCol by each component, component integration and full test functionality. However, due to the constraints mentioned in section 8.5, this test plan will be based on only unit tests.

1.5 Definitions and Acronyms

CI = Configuration item.

IEEE = Institute of Electrical and Electronic Engineers

QA = Quality assurance

CC = Cyclomatic Complexity

STP = Software Test Plan (this document)

TBD = to be Determined

2. TEST ITEMS

2.1 Program Modules

The program modules are:

- Client
- Common components
- Server
- Utilities
- Test Cases

The tester will be refactoring and executing unit tests on methods that give CC of 7 or higher in among above modules.

Developers will be responsible of writing additional test to correspond to the changes and running unit test against it.

2.2 User Procedures

The user documentation will be reviewed to verify against its usability by end users, and against the actual implementation to determine accuracy.

3. FEATURES TO BE TESTED

Client side, server side, utility functions and common work components are tested using Unit test cases that are in FreeCol project.

4. FEATURES NOT TO BE TESTED

UI components, were out of scope for this project.

5. APPROACH

5.1 Component Testing

Testers will perform a set of component testing and unit testing to ensure the components work as designed. All unit tests will be run after new components are integrated when merging each branch from GitHub repository.

5.2 Integration Testing

Integration testing will ensure that client, server and utility classes are integrated and working together. The test will be performed manually using a test matrix to verify the coverage, if time permits.

5.3 Interface Testing

Interface Testing is out of scope at this time.

5.4 Security Testing

Security testing will include (if time permits) attempting buffer overflow and cross-site scripting security holes to ensure FreeCol is not vulnerable to malicious attacks.

5.5 Performance Testing

Software games may require heavy computer resources to execute well. One of those resources is memory. Thus, testers will interact with FreeCol to test memory usage while running.

Initial performance testing will be to analyze the program while it is running. For this test we will use Google Performance Analyzer Tool. This tool will help detect any memory leaks that can occur during the running of the program. The results of this test can be found on the GitHub page related to the project.

Performance test (with time permitting) will be executed by using several test machines exposed under different stress level on normal game play while running the performance analyzer tool.

5.6 Regression Testing

Regression Test will be conducted by re-executing all test cases for all components related to the modified component when a change is made to the FreeCol to ensure that applied changes do not cause any adverse effect on previous tested functionality. With each check in of the code a unit test suite will be run to ensure that the product has not broken any key components of the software. Ideally we would want to run full regression testing on the software with each release, testing the release to determine if new code has effected any existing components, but full regression testing will not be implemented in the initial test plan.

5.7 Acceptance Testing

The initial requirements of the software are unknown so acceptance testing will be limited to include demonstrations for the team lead to inspect and provide feedback on the proposed changes. These changes must be implemented before the final release. Feedback will be in the form of verbal communication, via Google Hangouts, and via email as necessary. All major changes will be documented as appropriate.

5.8 Beta Testing

No beta testing will be performed at this time.

6. PASS / FAIL CRITERIA

6.1 Suspension Criteria

Test case execution will be suspended if a critical failure that hampers the ability or value in performing the associated test(s) is discovered.

6.2 Resumption Criteria

Test case execution will be resumed when a tester team member thinks the problem causing suspension has been fixed. All test cases that uses the modified portions of the project will be re-executed.

6.3 Approval Criteria

The results of each test case will be approved if the results meet the expected results description in the test case.

7. TESTING PROCESS

7.1 Test Deliverables

All deliverables will be in the shared repository at GitHub: Baymax2008/cosc603-Duffy-Encinas-Gale-finalproject. These deliverables include metrics reports and audits reports based on testing tools used.

7.2 Testing Tasks

- Download code and setup testing environments
- Review existing test cases
- Review and select testing tools to use
- Execute tests

- Report defects
- Complete test report

7.3 Responsibilities

All resources from this team are responsible for the completion of all components, final testing and deliverables.

7.4 Resources

The testing team has 3 members who work full time, are Computer Science graduate part-time students.

7.5 Schedule

The project schedule covers from the time this project was assigned (April 7) until is due (May 16).
Final selection and definition of the project to work: April 14.

The schedule will include assigned time frame for the tasks described in section 7.2 Testing Tasks.

8. ENVIRONMENTAL REQUIREMENTS

8.1 Hardware

- At least 256 MB memory although 512MB is recommended.
- PC or Mac using modern operating system
- Screen resolution of at least 1024x768 pixels.

8.2Software

- Windows 10, and Mac OS X.
- Java JDK 7 or greater
- Microsoft Office (Word, Excel) for reports and defect tracking
- Microsoft Internet Explorer 11
- Mozilla

- Eclipse
- Eclipse Memory Analyzer
- Google CodePro AnalytiX
- FindBugs

8.3 Security

The test environment doesn't have any specific security requirements.

8.4 Tools

The tools that will be used for testing FreeCol are:

Eclipse Memory Analyzer to find how much memory is consumed while the product is running. The results of this test will be contained in a zipped file called eclipsememoryanalyzerreport.zip and will be stored in GitHub under the main directory.

Google CodePro AnalytiX and its audits tools to perform static code analysis, remove defects and update empty statements red flagged based on the component audit. The audit file will be located in GitHub repository under metrics folder. File name will be audit violations .html

FreeCol documentation which describes how to install and running FreeCol. This documentation will be used to validate it guides the use of the game to the end user as intended.

Run all test cases that currently exist within the FreeCol code.

FindBugs that helps finding bugs in the code. The team will make appropriate changes that FindBugs recommends.

8.5 Risks and Assumptions

The test environment for this project is extremely constrained for resources. Constraints to this project are: members time availability – members work full time, are Computer Science graduate part-time students and are only available on weekends; selection of the project to be analyzed: finally decided on April 14.

An important risk is that the FreeCol is an open source software and anybody can commit to it. The Contingency plan to offset that risk is to have a moderator that accept and test any branches that are introduced before is introduced to the master branch.

An assumption the test team will make is that built- it test are working precisely in target testing.

9. CHANGE MANAGEMENT PROCEDURES

The procedure for changing the test plan is as follows:

Propose changes to FreeCol using the shared repository at GitHub: Baymax2008/cosc603-Duffy-Encinas-Gale-finalproject and using the branches approach the GitHub provides.

The team will discuss the proposal and either reject it, accept it, or accept it with modifications. If the team accepts the proposal, then it and any agreed upon modifications will be implemented.