**Anthony Miller**

**Zachary Sierakowski**

**Garrett Steele**



**Apotheosis Tactical Role-Playing Game (TRPG)**

**Software Quality Assurance Plan**

**Document History and Distribution**

1. Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision #** | **Revision Date** | **Description of Change** | **Author** |
| 0.1 | 2015-05-12 | Defined packages to be tested | Miller, Sierakowski |
| 0.2 | 2015-05-16 | Laid out testing goals, tasks, etc. | Miller, Sierakowski |
| 1.0 | 2015-05-19 | Final revision before submission | Miller |
| 1.0.1 | 2015-05-19 | Added unit test tool | Miller |
|  |  |  |  |
|  |  |  |  |

**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

**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 described testing activities will be completed on an existing project which was described by the original developer as in alpha status. A cursory investigation showed that it was both buildable and runnable, which were main criterion for project selection.

As determined by Google CodePro, the two packages fer.ai and fer.gameplay have the most internal dependencies, mostly on each other, and fer.ai has a high average cyclomatic complexity. These packages will be the focus of testing and refactoring.

Using testing tools, we find and correct design smells, refactor, and develop unit tests. We will attempt to correct any bugs that are found as a result of testing.

**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.**

**1.2a Design Smells**

Purpose: To find poor program designs based on an object oriented approach to programming.

Items to be Tested: packages fer.ai and fer.gameplay

Technical Approach: Tools to be used include *JDeodorant, Checkstyles,* and *CodeCity*

Roles/Responsibilities: 1.2a and 1.2b can be completed concurrently

**1.2b Refactoring**

Purpose: To correct poor design and programming to increase readability, efficiency, and reflect modern conventions.

Items to be Tested: packages fer.ai and fer.gameplay

Technical Approach: Both the built in refactoring capabilities of the Eclipse IDE, *AutoRefactor, JAutoDoc,* and *SpartanRefactoring* will be used

Roles/Responsibilities: 1.2a and 1.2b can be completed concurrently. Refactoring should include code documentation to help explain changes.

**1.2c Unit Tests**

Purpose: To develop tests for each function of program code that can be used to determine if the code is operating as intended under a myriad of conditions.

Items to be Tested: packages fer.ai and fer.gameplay

Technical Approach: To cover as much code base as possible, JUnit tests will be generated/modified by MoreUnit and/or Google CodePro and modified by hand when necessary.

Roles:Responsibilities: 1.2c should only be completed on code that has passed 1.2b

Pass/Fail: Before the end of the development period, all unit tests should pass during runtime.

**1.3 Scope**

This test plan focuses on a snapshot of a project. No new functionality is being added during this testing cycle.

**1.5 Definitions and Acronyms**

**Alpha** (project status) - describes when a project is still in early stages of development; it may run but still includes many known bugs and/or lack of complete functionality.

**Cyclomatic Complexity** - a quantitative measure of the number of linearly independent paths through a program's source code

**Design Smells** - any symptom in the source code of a program that possibly indicates a deeper problem with program design

**Object Oriented Programming** (OOD) - a programming language model organized around objects rather than "actions" and data rather than logic.

**Refactoring** - the process of improving the internal structure (both design and code) of software without altering its external behavior.

**Unit Tests** - a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use.

# TEST ITEMS

*(Specify the test items included in the plan. Supply references to the following item documentation:*

* *Requirements specification - Not available*
* *Design specification - Not available*
* *Users guide - Not Available*
* *Operations guide - Not Available*
* *Installation guide - Not Available*
* *Features (availability, response time) - Not Available*
* *Defect removal procedures - if found, defects are to be corrected by the tester*
* *Verification and validation plans - Since no requirements or design specifications are available, the only verification and validation is whether it compiles and whether it runs*

**2.1 Program Modules**

*Reason for omission: There are no new module development occurring during this test period.*

**2.2 User Procedures**

*Reason for omission: There are no existing user usage guidelines*

# 3. FEATURES TO BE TESTED

Project Design - limited to packages fer.ai and fer.gameplay because of analysis of dependencies and time constraints.

Unit Tests - limtied to packages fer.ai and fer.gameplay because of analysis of dependencies and time constraints.

# 4. FEATURES NOT TO BE TESTED

Game functionality itself is not to be tested because the game is in alpha and many features have not been implemented by the developer yet.

# 5. APPROACH

**Prior** to start of testing, the project was ported to the Eclipse IDE to allow use of tools designed for Eclipse. Additionally, the project was modified to be able to compile on operating systems other than Unix, such as Windows.

**5.1 Component Testing**

The packages fer.ai and fer.gameplay are the focus of the testing. a multitude of Eclipse plugin tools will be used to analyze the design and style, help refactor, and develop unit tests.

The tools used for style and design are:

JDeodorant - functions include GodClass, LongMethod, TypeChecking, and FeatureEnvy

CheckStyles - Alerts and can correct tester of some style issues

CodeCity - Builds a visual ‘city’ of the code with stats and metrics

The tools used for refactoring are:

AutoRefactor - Used to convert some normal comments to Javadoc style

JAutoDoc - Used to analyze and autogenerate Javadoc style comments

SpartanRefactoring - Used to help simplify conditional statement ladders

The tools used for unit tests are:

MoreUnit – Used to generate Junit Tests

Google CodePro – Used to generate Junit Tests

Additionally, Google CodePro was used to analyze dependencies to focus efforts, and FindBugs was used to check for easy to find bugs.

**5.2 Integration Testing**

*Reason for omission: Testing only snapshot of project; no new functionality or components will be added during test cycle.*

**5.3 Interface Testing**

*Reason for omission: Testing is focusing on packages that do not affect the graphical user interface*

**5.4 Security Testing**

*Reason for omission: Testing is focusing on packages that do not use network or storage resources.*

**5.5 Performance Testing**

*Reason for omission: The snapshot of the project is still in alpha status and still would have much that needs to be developed. Testing the performance of what is currently available would be useless since many components may change*

**5.6 Regression Testing**

To test that refactors did not break the project, it should be compiled and verified in a runnable state before any changes are committed to the repository.

**5.7 Acceptance Testing**

*Reason for omission: The project does not have any clearly defined requirements or goals.*

**5.8 Beta Testing**

*Reason for omission: The project does not have a clearly defined customer. Additionally, testing is only being done on a snapshot of alpha code.*

# 6. PASS / FAIL CRITERIA

**6.1 Suspension Criteria**

If the laptop runs out of fuel, suspend tests.

**6.2 Resumption Criteria**

Tests may resume after refueling.

**6.3 Approval Criteria**

Each testing section (1.2a, 1.2b, 1.2c) only requires approval from the tester assigned to that section.

# 7. TESTING PROCESS

**7.1 Test Deliverables**

Apotheosis-TRPG project GitHub repository

**7.2 Testing Tasks**

Install Eclipse IDE

Install Eclipse plugins listed in 8.4

Checkout project with SVN URL: <https://github.com/Sustenance/Apotheosis-TRPG>

Complete 1.2a and 1.2b

After design is stable, complete 1.2c

**7.3 Responsibilities**

Testers: Anthony Miller, Zachary Sierakowski, Garrett Steele

**7.4 Resources**

Testers:

* Anthony Miller
  + Maintain Test Plan Document
  + Configure and manage repository
  + Port project to Eclipse and Windows
  + Assist in code design analysis and refactoring as needed
* Zachary Sierakowski
  + Analyze project code for design issues
  + Refactor project to correct design and/or bugs
  + Assist in unit test generation as needed
* Garrett Steele
  + Generate unit tests for all classes in fer.ai and fer.gameplay
  + Correct unit tests and/or project code to ensure tests pass
  + Assist in code design analysis and refactoring as needed

**7.5 Schedule**

*Reason for omission: No clearly defined test schedule.*

# 8. ENVIRONMENTAL REQUIREMENTS

**8.1 Hardware**

Necessary: Computer capable of running an operating system that has a JDK 1.8 available for it and a graphical user interface.

Desired: Computer with powerful processor and large amounts of RAM

**8.2 Software**

Necessary: Operating System with a graphical user interface, Java Development Kit version 1.8, Eclipse IDE

**8.3 Security**

*No security measures are being considered during this phase of testing.*

**8.4 Tools**

JDeodorant Used to identify- poor design in software code.

Google CodePro - Used to help analyze dependencies and metrics of code, and generate Junit tests

MoreUnit – Used to generate Junit Tests

JAutoDoc - Used to generate Javadoc style comments

CheckStyles - Used to check code style

AutoRefactor - Used to auto modify comments

SpartanRefactoring - Used to modify conditional statements

CodeCity - Used to analyze and visualize code stats and metrics

FindBugs - Used to search for simple bugs in code

**8.5 Risks and Assumptions**

Time is a critically short resource for this phase of testing. Because of that, proper coordination between testers is more difficult and there runs a risk of duplication of efforts or conflict of efforts.

# 9. CHANGE MANAGEMENT PROCEDURES

*Reason for omission: No real change management procedures are in place. If a change is needed, it is made without much discussion.*