Software Testing Plan

Maxwell Sweikart, Nina Yakovchits, Ryan Castner

April 2015

1 Introduction

In this brief document we will discuss the testing plan and implementation that we have decided to use to complete the task of refactoring an existing codebase and making it as extensible as possible.

Our approach is to utilize a full suite of JUnit tests that modularize the codebase and examine all applicable methods to ensure that given an input, a specified output is achieved. In this way we will ensure the spirit of refactoring is maintained by not changing the external behavior of the code, for instance API calls or method calls that might be used by legacy systems, yet still allow for cleaner and more readable code that is more extensible to future applications and domains.

2 Test Items (Functions)

The following is a list of the class files to be tested by our JUnit test suite:

1. CreateDDLMySQL.java
2. EdgeConnector.java
3. EdgeConvertFileParser.java
4. EdgeConvertGUI.java
5. EdgeField.java
6. EdgeTable.java
7. ExampleFileFilter.java

We will also update the suite to test any new classes that may come into play during the refactoring process as well as including different inputs to prove extensibility of the code.

3 Critical Functions

Here we will enumerate and expand upon the critical functions of each of these listed class files that must pass on each build of the project. If the JUnit tests for these cases specifically fail the code should not be pushed to production until it has been rectified.

CreateDDLMySQL

1. testGetDatabaseName
2. testGetSQLString
3. testGenerateDatabaseName

EdgeConnector

1. testGet* methods
2. testSet* methods

EdgeConvertFileParser

1. testEdgeConvertFileParser
2. testOpenFile

EdgeConvertGUI

1. testEdgeConvertGUI
2. testShowGUI

EdgeField

1. testEdgeField
2. testGet* methods
3. testSet* methods

EdgeTable

1. testEdgeTable
2. testMoveFieldUp
3. testMoveFieldDown
4. testMakeArrays

ExampleFileFilter

1. testExampleFileFilter* methods
2. testAcceptFile
3. testAddExtension
4. testGetExtension

4 Approach (Strategy)

The overall strategy will be to use the built-in JUnit test suite functionality of the Eclipse or NetBeans IDEs to perform and check the results of the JUnit test suite. Options like ANT builds were considered but will not be used due to team preferences. Github will be used as a collaboration and version control tool for source code refactoring and for test code implementations. Any updates to functionality due to refactoring must be reflected with appropriate updates to test cases or implementations of new test cases to verify these behaviors. Furthermore, no existing tests should be broken or have to be modified due to functionality updates as that would indicate a divergence from refactoring, which is beyond the scope of our task. This document will be updated to include new test items and critical functions that must be passed in future build versions as well. These changes will be recorded by github, where this document will be hosted.