Team Information Final Report

Martus Test Suite

Miki Sugimoto
Donovan Roseau
Mac Knight

Table of Contents

Chapter 1: Project Selection

Our team has decided to use Martus because of how the open-source project allowed individuals in countries that are experiencing war or any other type of conflict to report firsthand what they see. It is also written in a language, Java, each of our group members are familiar with, so that makes it easier to read and test code. In the beginning, we attempted to run the desktop version of Martus but ran into difficulty when trying to download eclipse on a virtual machine. Operating Eclipse was time consuming due to it "moving slow" and not compiling the code since Martus was created on an older version of Eclipse. We also considered using an older version of Martus as our project since we were able to get it to run, however, we could not find the source code for it.

We decided to change from the desktop to the android app version of Martus since the desktop version was to difficult to compile. The project has a directory with instructions to set up and run the app in the their repository. The instruction were straightforward as what the user should do. First we had to download the Android Studios onto Linux. One of our teammates ran into a small problem at this point and had to install a few libraries in order to finish the download. Here are the lines of code if you encounter the same problem: sudo apt-get install lib32z1 lib32ncurses5 lib32stdc++6. After, download and import the source code from the Martus repository. The code ran smoothly afterwards, but Android would freeze from time to time.

After sometime exploring the project we were able to find some gradle scripts that would run several tests for the project, and return in the event log whether or not certain sections of tests passed. While we were able to find the individual classes that would test a certain condition, we have not been able to run them individually and record the results. One of the problems that would arise when trying to run some tests individually without gradle was a runtime error with the message “Stub!”.

Chapter 2: Test Plan

Introduction

For this week’s deliverable, the team dove into the task of testing code written by another developer (Martus). The lack of documentation on the methods was a setback to producing our first 5 test cases, but we managed to find some simple methods that should be the easiest to assess for our first experience with the process.

Requirements

Although the mission of the project is stated on the website, we have been unable to find any concrete requirements or expectations as to how the app should perform, or what features, etc., the developers are hoping to eventually add. This is not critical to the testing we will be performing, but it also leaves us without many clues as to how the different features are supposed to function.

Test Cases

Test Name: testOnOptionsItemsSelected
Method: onOptionsItemsSelected
Input: Menu Item
Object Output: boolean

Test Name: testOnCreateOptionsMenu
Method: onCreateOptions
Menu Input: Menu Object
Output: boolean

Test Name: testGetSettings
Method: getSettings
Input: None
Output: mProgress Dialog Handler Object

Test Name: testGetBackButton
Method: getBackButton
Input: None
Output: Image backButton

Test Name: testGetApplicationName
Method: getApplicationName
Input: None
Output: String

Testing Schedule

9/27	Create and specify five of the eventual twenty five test cases that we will eventually test.
10/18	Rework test plan if necessary and create a framework to implement test plan.
11/10	Framework should be completed and twentyfive cases should be ready for testing.
11/22	Use five faults into code that will cause tests to fail.

Constraints

The team has already encountered several restraints and expects several more before the testing is complete. The primary concern is that the source code has little to no documentation as to what the methods are supposed to do, and several take in objects which are equally undocumented. This has presented a challenge in trying to figure out how to create tests for these methods. Another challenge has presented itself in that we are unable to run one test individually and examine the output, therefore adding another layer of confusion in creating our own tests for the methods.

The software also runs on very outdated on versions of all of the build tools and environments, even to the point where it will likely not function correctly on a current Android device. This is creating a very slow and buggy process, especially when it is required that the software interact with any new versions of the software, which means we are unable to test the product on an actual phone, which would be a helpful resource.

System Tests

Full system test are almost impossible at the moment, as the server connection with the emulator is not working. As mentioned previously, the app is also not allowing access to the site on a real phone, so it is unknown to us at this time if it even works in the “real” environment.

Chapter 3: Testing Framework

Introduction

For this week’s deliverable, the team dove into the task of testing code written by another developer (Martus) and developing drivers which test methods within a class. The lack of documentation on the methods was a setback to producing our first 5 test cases, but we managed to find some simple methods that should be the easiest to assess for our first experience with the process. We have significantly fleshed out our test case template since the last deliverable and it is now ready to be read by a script.

Test Cases

Test ID: 001
Requirement Tested: The method should return an integer, that shows the number of days between date1 and date2
Component Being Tested: Flexidate(date1, date2)
Method Being Tested: getRange()
Test Input: date1 =2001,5,5 date2 = 2002,5,5
Expected Outcome: 365

Test ID: 002
Requirement Tested: The method should return an integer, that shows the number of days between date1 and date2
Component Being Tested: Flexidate(date1, date2)
Method Being Tested: getRange()
Test Input: date1 =1999,5,5 date2 = 2000,5,5
Expected Outcome: 366

Test ID: 003
Requirement Tested: The method should return an integer, that shows the number of days between date1 and date2
Component Being Tested: Flexidate(date1, date2)
Method Being Tested: getRange()

Test Input: date1 = 2020,4,25 date2 = 2020,5,25
Expected Outcome: 31

Test ID: 004
Requirement Tested: The method should return an integer, that shows the number of days between date1 and date2
Component Being Tested: Flexidate(date1,date2)
Method Being Tested: getRange()
Test Input: date1 = 1999,12,31 date2 = 2000,1,1
Expected Outcome: 1

Test ID: 005
Requirement Tested: The method should return an integer, that shows the number of days between date1 and date2
Component Being Tested: Flexidate(date1,date2)
Method Being Tested: getRange()
Test Input: date2 = 1776,6,4 date2 = 1776,6,4
Expected Outcome: 0

Report

For our framework, we developed a driver for each class in Martus. The method we are currently testing is getRange() and it is found inside the Flexidate class. Our driver has a main and when it executes, an object is instantiated and the method is executed. In order to test the getRange() method, two Calendar objects must be created and given three integers which serve as dates on a calendar. Then a Flexidate obeject is instantiated and takes two parameters which are two of the dates just created. Next, the getRange() method is called on the Flexidate object and the number of days between the two dates are returned as an integer. The results are sent to the temp folder and compared to the results in the oracle folder.

The script first runs the driver tests and the results go into text files in the test results folder. Then, the driver compares the outcomes of the tests to the oracle and records the results in a new file and displays them in the default browser.

In order to run the test from the command line, change the directory to the scripts folder inside of TestAutomation_1 and write chmod u+x RunAllScripts to get permissions. Then, ./RunAllTests so the driver is executed and the results are sent to the temp folder.

Chapter 4: Full Test Suite

Introduction

For this week's deliverable, our team had to create 25 test cases and have our script read the test case files, compare the test results to the oracle, and have the outcome of the comparison open in a html file.

Process

For our framework, we had trouble finding classes that we could test since the Martus Android app is not complete, and has less testable classes than the desktop version. However, we found two testable classes which were the Flexidate class and the Comparison class. From these two classes, we obtained our 25 test cases. In the Flexidate class we used methods such as getRange, getDay, getYear, and getMonth. And inside the Comparison class, we used methods like compare(), safeComepare(), and rangeOverlap.

For the runAllTest script we ran into some trouble reading the oracle file and comparing it to the result files. We discovered later that the new line character was critical to the script parsing the information correctly and passing it along to the driver. We made a big improvement from the last deliverable because now our script reads from the testCase files one by one and calls the correlating testDriver passing in the correct information instead of having the input hardcoded.

Report

Scripting has been one of our hardest challenges for this deliverable but we feel confident about getting it complete before the deadline. We have been able to pick up more and more scripting skills as the project has progressed and now feel confident with file reading, loops, assigning variables, and navigating file trees from within the script. We also had a significant amount of CSS and HTML formatting in our final report display, which gave us the opportunity to refine those skills as well as learn the means to concatenate large chunks of text to a file instead of typing the write command for each line we wanted to enter. In the end, our table is easy to read quickly and discern which tests have passed and failed.

Finding 25 test cases was also a huge problem but we managed to get them all. Our original plan was to test methods which accepted objects as input, but in the end the goal was to work on our ability to build a test suite and not manipulate Martus' poorly documented code, so we opted for simpler methods for our purposes.

Chapter 5: Fault Injection

Introduction

For this week's deliverable, the team dove into the task of testing code written by another developer (Martus) and injecting five faults into the project's source code. The injections were done in order to see if it altered the outcome of our twenty-five test cases. Hopefully, when a fault is injected, it will not change the outcome of each test case, only the the methods that he fault is targeting.

getYear() method

The first change we made was to the getYear() method. In order to duplicate this injection, follow the class path TestAutomation_1/project/src/org/hrvd/util/date/Flexidate.java. Inside the Flexidate class change the operator inside the getYear() method from date / YEAR_MULTIPLIER to date * YEAR_MULTIPLIER.

getMonth() method

______The next change we made was to the getMonth() method which returns the month from a date. Follow the path TestAutomation_1/project/src/org/hrvd/util/date/Flexidate.java. Inside the Flexidate class change the operator insid the getMonth() method from `(date % YEAR_MULTIPLIER) / MONTH_MULTIPLIER` to `(date % YEAR_MULTIPLIER) * MONTH_MULTIPLIER`

getDay() method

______Next we injected a fault into the getDay() method which return the day from a flexidate. In order to inject a fault follow TestAutomation_1/project/src/org/hrvd/util/date/Flexidate.java. Inside the Flexidate class change the operator inside the getDay() method from `date % MONTH_MULTIPLIER` to `date * MONTH_MULTIPLIER`.

getRange() method

______Follow the class path like the previous methods in order to get to the getRange method. The method returns the difference between two dates.

getRangeContext() method

______Inside Flexidate.java go to the getRangeContext() method we change the if statements. For example, we changed `if ( getDay() != 0 ) return DAY_CONTEXT` to `if ( getDay() == 0 ) return DAY_CONTEXT`.

Chapter 6: Project Experiences

Experiences

Overall, this project was a challenging but rewarding experience. We had to learn independently how to script, create test drivers, and create a testing framework. At times these challenges seemed daunting and we were not always confident in our ability to complete a task on time, but in the end we produced a functional and effective product. We experienced a few stumbles; some were our own problems, i.e., time management, and some were due to poor documentation and work by the teams which produced the Martus software. However, we learned valuable lessons about communication and expectations of ourselves and our team members through these mishaps and ended up coming out with a much better understanding of team dynamics. We also learned about the importance of documentation, particularly on a large project, and why professors really drill it into us from the beginning. It was at times nearly impossible to understand what the original intent of code was, and for that reason we nearly abandoned the project in favor of another, more coherent one. This software engineering project truly did teach us what it means to produce a good product every step of the way.

Self-Evaluation

Our team did solid work throughout the project with each member contributing something new each step of the way. Although we did stumble and were not prepared completely for a small number of the deliverables, we did rectify the problems that arose almost immediately and consistently worked towards the next goal. Team members were flexible and amenable to meeting and working together, and it didn’t ever feel that one person was shouldering all of the work at any point. We believe our final product is a reflection of our collaboration and the application of our strengths to produce our best work.

Project Assignments

While we did eventually complete all necessary sections of the project, we were a little thrown by some of the deliverables. It seemed as though some were fairly easy, such as picking test cases, whereas others, like getting the script working with the files, were enormous undertakings. It might have been helpful to have steps such as those broken into smaller increments, or have more practice assignments like the HTML script file to help guide us towards the finished product.