Team Soft Boys: Deliverable 3

Joe Ayers, Maz Little, Jasmine Mai, Joe Spencer

Introduction

Our team is building a testing framework for sugarlabs, an HFOSS project that consists of a core operating system, with official and crowd-sourced educational “activities” that users of the system can access from the operating system. Currently, we have a framework that can run specified inputs and output print statements regarding the status of the test to an unformatted html file that opens automatically. The framework script locates the testing report file, empties it, navigates to the test case folder, runs the test case files in the folder, outputs each result to the HTML file, and opens it. The code is shown below:

```
#either find or create testing report

touch ../reports/testingreport.html

#empty testing report

echo ''>../reports/testingreport.html

#change directory to test cases

cd ../testCases

#run each python file, change system output to test report

for entry in ./*

do

python $entry >> ../reports/testingreport.html

done

#open testing report

xdg-open ../reports/testingreport.html
```

We currently have our test cases hard coded into another python file to allow us to make sure our test frame works; the file loops through a pre-defined list of inputs and feeds them to the above code (rather than opening individual files). Our next steps with this script are to define (twenty more) individual inputs and refine the script so that it all runs automatically.

Progress

Problems

We've run into a few problems over the course of our project so far. Three in particular have affected this deliverable. First, the way sugarlabs is setup is riddled with dependencies. This compounds another problem: there are several libraries that need to be imported for some of the functions we're testing, but some of the imports weren't recognized when we ran the script. Finally, on the more conceptual level it has been difficult to decide exactly what we want to attempt to test about sugar labs, since it is an entire operating system. We can't (and aren't) aiming to produce an entirely comprehensive testing frame, so we've decided to focus on the core sugar operating system and its data and functions. To that end, we've been reworking our test cases as we are able to learn more about sugar and what's reasonably testable for our team and project constraints.

Solutions and Next Steps

However, thus far we've been able to mitigate these problems. Some of the dependency problems have been avoided by moving from a Mac environment to an Ubuntu environment on VirtualBox, particularly because we can set up the dependencies in a clean environment (i.e., there's nothing else installed besides the out-of-the-box applications. Next, we also reconsidered the methods we were testing, and currently are testing methods that don't require the import dependencies that are throwing errors. We considered creating dummy versions of the files that the program is trying to import, and while we haven't done so yet, we're going to attempt to do so in order to expand our test cases from the five we have functional to the twenty-five we need.

Reactions and Impressions

Thus far in the project, we feel good with what we have, but aren't entirely satisfied with the complications that have arisen from this project. It was never going to be easy, but as with all projects requiring a product someone else created, the way it's set up is less straightforward than would be desirable (particularly dealing with the dependencies). Additionally, because of the complexity of testing and the system, it's hard to walk through both the system and what we want to test manually to see if our testing is effective. HFOSS projects tend to be either very large or very small in scope, and we picked a project in the former category. Our original project, Tanaguru, was too small. We were really only testing one algorithm that took in one int and output another, with outputs that were difficult. This one, as previously mentioned is an entire operating system.

Test Case Choice

We ended up choosing (and refining our choices of) test cases for this project based on the above problems and constraints we've run into. In particular, we wanted to make sure we stayed within the scope of what we originally intended for our project. To that end, we have chosen to test just the core operating system, and avoid testing any of the activities to make the scope of work managable for four people devoting a small part of their schedules to the project. We've found it to be a reasonable constraint to plan for to avoid overcommitting.