Joseph Silva Jr.

11/28/2020

CS 320 Journal 5-2

SNHU

* **What were the software testing techniques that you employed for each of the milestones? Describe their characteristics using specific details.**

The following is a list of software testing techniques that I have employed during each of the milestones: static, dynamic, automated, functionality, white-box testing, and unit. While using Eclipse, I was able to use static and dynamic testing techniques. In the edit area of eclipse, I was notified if I had any errors or warning withing each class of code. This would be static testing because I know about the errors/failures in my code structure without having to execute a testing program. The Junit Tests allowed me to use dynamic testing technique when executing the code to see if there were any failures/errors after the test classes tested my class files.

Next, the automated testing was when I used Junit Tests for the Contact files, Task files, and Appointment files. I also used the functionality test while using Junit Tests because they tested each file without implementing these files into a full program. The two-last type of testing technique was unit testing and white-box testing. I used these two types of testing techniques during each milestone. Each class file is being tested separately as a unit to make sure they individually work without any errors or failures. Unit testing is a form of white-box testing because my code is going through a workflow to make sure everything in the individual class does not conflict with each other.

* **What are the other software testing techniques that you did not use for the milestones? Describe their characteristics using specific details.**

The following is a list of software testing techniques that I have not used in the milestones: manual, integration, system, and acceptance. I did not perform any manual tests regardless of me writing the test class files myself because I was able to configure a test program while using the Junit Testing technique. The next two, integration and system, were not used because I was testing each class as an individual file and not the compatibility with each other. I have not tested a milestone’s files with another milestone’s files to make sure files such as Contact.java and Task.java will work together. Since I have not used these programs together, I did not build a main structure to test these files at an entire program. The last type of testing I did not use was acceptance because I need a full program to make sure it meets a user requirement, but since there is no main class file, I am unable to conduct this type of test.

* **For each of the techniques you discussed, explain the practical uses and implications for different software development projects and situations.**

All the techniques are very important when development a software project. Techniques like static and dynamic are used throughout the entire project especially when you are writing code on a program like Eclipse because these techniques help prevent any errors and/or failures while writing the code. Automated and Manual testing can be implemented when it comes to the amount of input you want to test to make sure it’s the expected output. If we are working with a few possible inputs, manual testing can be done using a small amount of time, but if we are trying to test thousands of inputs automated tests will save a tester time by having the program conduct the tests for you. The rest of tests such as unit, integration, and structure tests are implemented to make sure there is no error throughout the entire program. I compare these types of tests to real-world examples such as a workforce. We start with unit testing to make sure each class file works properly on its own. This type of test can be compared to an individual law enforcement officer. We run tests on the individual law enforcement officer to make sure they are physically, mentally, emotionally, and intellectually fit for the job. The next is integration tests make sure each file works together and notifies the testers of any conflicts between these files. I compare this to a law enforcement work shift because we have to ask does every member of this shift work well with each other as a team. The structure tests are the testing of the entire program has a whole to make sure there are no errors/failures as an entity. I compare this type of testing as a law enforcement station or department.