Lars Dela Cruz

CS-320: Software Test Automation

Southern New Hampshire University

**7-2: Project Two Summary and Reflections**

The main approach with the software requirements given was to fulfill each specification with code that is accompanied by Junit tests that confirm its integrity. Lets take for example our contact class and task class. The Junit test assists in accomplishing the requirement of a unique ID not exceeding 10 characters by using a function that checks on the length of an input.

As weeks go by and we continue on with the modules, the coverage percentages also continue to improve. This coverage percentage is direct feedback into code at least functioning and as it goes up you may see the progress in the actual quality of the program. To keep things “technically sound” the contact class in the line of “CONTACT = new ArrayList();” helps in both new additions and deletion of the items in our list along with the Junit tests. Stack overflow, youtube and simple google searches helped immensely in the process of creating the program but with JUnit tests, most of development was solved through trial and error. Efficiency in the code came through when constant testing/running with a trial and error procedure was put in place.

Frequent successful testing that is kept organized so the Junit tests match our classes was a technique that advised proper software testing methods. This step by step procedure applied separate focus on each need, for the areas in logic and structural conditions Junit test with a White box approach helped a lot. Since I had constant compilation and running of code to find any issues, I did not follow a static testing/black box process.

Static testing is a hind sight method of observation in reviewing the code without running it. If development was very early on, I may have used static testing. Software testing is assisted by automation like Junit tests but this role does not reward shortcuts to be taken. The testing is automated but careful review of the output of the program with the client needs put in place is the main goal.

Code written by myself could very well be met with bias. To limit that bias an open mind that is open to change can assist in the refusal of alternatives. Mentioned before, we are working with automated testing instead of doing it manually. In this case, It is very easy to see yourself as better than you actually are, especially in Junit tests that assess baseline functionality and not necessarily real quality. Discipline in software testing is important for the project to face as little amount of bugs as possible. If a sea of glitches appear late into development, and they should've been caught earlier, this could mean a catastrophic failure in not only program functionality but company costs as well.

Discipline does not mean that work is just done and over with but how that work is done with the quality of focus applied, no matter how much time you have labored for. Cutting corners also reflect the level of competence as a developer to your employer. Not only is your appearance in surveillance but the development team as a whole, since projects represent ALL of their work.

***Sources***

Sheeta (2023) *Cognitive bias in software testing: Why do testers miss bugs?*, *Software Testing Help*. Available at: https://www.softwaretestinghelp.com/cognitive-bias-in-software-testing/ (Accessed: April 16, 2023).

Farias, H. (2018) *Where you should never cut corners in software development*, *InfoWorld*. InfoWorld. Available at: https://www.infoworld.com/article/3286265/where-you-should-never-cut-corners-in-software-development.html (Accessed: April 16, 2023).