Summary and Reflection

Sergio Mateos

Southern New Hampshire University

Summary and Reflection

One of the important parts of the Software Development Life Cycle is the testing phase where developers test and run the code or program to verify that meet requirements according to customer criteria and developer are available to fix any bug. Developers approach the problem using different testing techniques to meet customer requirements. In the project, I was a software engineer for Grand Strand System, my goal was to incorporate the code and unit test for a mobile application. The development of the code needs to have a Contact Service, Task, and Appointment Task anther respective classes.

First, I decide to create each package independently to have control over the testing, also the organization was a huge factor in my development of the code. After I create the first class, I decide to create a testing package where the JUnit test will verify if the code is working or not. Each class has different requirements, but the approach was slightly similar. I ensure that each class was meet the requirement, and I started verifying if that was true with the JUnit test approach. The JUnit tests verify that our code was working and Runs 30/30 which means that every line was verified and approved. There was no error in any part of the code and the running time was calculated as 0.215 seconds which is fast enough. The coverage is the number of lines of code that are successfully validated. The testing was effective to cover the requirement, but project coverage was 84.5% overall. For example, ContactServiceTest.java with 65.7% has the lowest testing coverage which means that the number of lines in this class is not validated compared to AppoinmentServiceTest.java with a 96.7%.

The code must be easier to understand for further updates, one of the ways that I used to make sound technically is each test has @Test annotation above and a brief description of what is intended to do while testing. @Test is allowed to be executed by import org.junit.api.Assertion. For example:

@Test

// Test to Add Contact with a unique ID

**void** testContactServiceClassAddWithId() {

ContactService service = **new** ContactService();

service.addContact("1234", "Enrique", "Zarate", "9999999999", "21527 E Homstead");

Assertions.*assertThrows*(IllegalArgumentException.**class**, () -> {

service.addContact("1234", "Enrique", "Zarate", "9999999999", "21527 E Homestead");

});

}

The efficient and easy way to ensure that the code was efficient is by using short words, understandable names, and comments that describe the purpose of each section in the code. If we work in teams, these methods will allow any team member to fix or update the code in case the developer is not there. Even if the developer is working by himself, comments on these techniques can refresh the mind of the developer in case of any further updates. For example:

//Verify requirements for the ID, First Name, Last Name, Phone Number ,and Address

`**public** Contact (String id, String firstName, String lastName, String phoneNumber, String address){

**if**(id == **null** || id.length() > 10){

**throw** **new** IllegalArgumentException("Invalid ID");

}

Every developer is different, and they have different ways to approach and develop new projects. Developers have been using different ways to test which becomes more efficient when testing a program. There are many software testing techniques, but the testing technique that I employed was the white box. White box, also known as structure-based testing, allows the testers to inspect and verify the auctionability of the code, infrastructure, and integration. One of the important of White Box is the statement coverage which allows the developer to identify where the code needs to be fixed. White Box provides input to exanimate the outputs. Also, White Box technique avoids repetitive statements.

White Box was an efficient technique in my development of the code, but there are many other software testing techniques like static testing techniques. Static Testing Technique is a type of testing technique where the test elements are not executed, instead involves manual functional testing processes, which becomes harder for testers o find errors without running the code. Static Testing is usually used at the early stage of development to avoid any error, and in case there is one it’s easier to fix. One of the usage scenarios where the Static Testing Technique is effective is on data flow analysis, tester will follow the path where the data is being used and analyzed to transform into output.

Learning something new can be hard and sometimes scary, but the right mindset and attitude will let you succeed in any problem. I had a small experience dealing with testing programs or software, and this limited knowledge make me unsecured in my results, but as I progress in the class my knowledge and security when developing a new code increased. The approach that I took while developing Project One was motivated and challenging since it was implementing a whole new program combining the last three assignments but fixing problems from the last assignment and ensuring that it can be run all at the same time successfully. Every part of the program was detailed and analyzed to prevent error, even though I created the package before, I decide to have been more careful in case it can be run all at the same time.

Since the project is a major assignment, I like to pay extra attention and prevent any bias that I can face. One of the strategies I implemented is the code little in this case was one class and then tested. I created the JUnit test as develop classes. Also, once I add testing JUnit test, I run them all at the same time to verify that there are no problems with the program. My responsibility of delivering a useful code that meets customer requirements can bring some stress and, in some cases, people tend to procrastinate, but setting time and focus will develop a better code.

One of the skills that everyone must work on is discipline. Discipline will create constancy and efficiency in any goal possible. In our scenario, discipline makes me follow a schedule and a process when developing my code. If I didn’t have the organization and dedication that I put in every day, probably my code would fail, or it could be incomplete.