7-2 Journal: Project Two

Richard Howell

CS-320

SNHU

2/16/2023

1. **Summary**

* *Describe your unit testing approach for each of the three features.*
  + *To what extent was your approach aligned to the software requirements? Support your claims with specific evidence.*

Unit testing as a whole is extremely important in the software development lifecycle (SDLC). Testing allows developers to reduce the risk of errors during development and improve the quality of the code, saving time, money, and resources. For each of the three features covered in project one (Appointment Service, Contact Service, Task Service) I applied both black box and white box testing techniques that aligned directly with the software requirements. A specific example of this is shown within the Appointment.java class. A string needed to be created that had a specific ID which could not be null and no more than ten characters, a date which could not be null and not set in the past, and a description that could not be null with no more than 50 characters. The tests I developed for this specifically addressed each one of the user requirements by testing both parameters that met the requirements and parameters that did not meet the requirements. The parameters that met the requirements were shown to assert to true when tested and the parameter that did not meet the requirements, such as null or too many characters, were shown to throw an illegal argument exception. This shows that my testing was directly aligned with the user requirements. I repeated this testing technique for each of the classes I was instructed to implement. The overall process for testing I implemented for each class was essentially the same. I verified that for all classes developed correct parameters asserted to be true and that incorrect Graphical user interface, text, application

Description automatically generatedparameters threw an illegal argument exception.

* + *Defend the overall quality of your JUnit tests. In other words, how do you know your JUnit tests were effective based on the coverage percentage?*

I believe that the overall quality of my Junit test were adequate based on my overall coverage percentage of my code was 85.4 percent. There are some improvements that I could make as I gain more experience. I believe that the reason I do not have 100 percent code coverage is due to my inexperience writing Junit test. As I gain more experience, I will be able to work out some of those inefficiencies. Even though my code coverage was only 85.4 percent, all the user requirements were tested with the test cases developed and were insured to be working correctly. Based on that I believe that 85.4 percent coverage is more than adequate for the project that I developed.

1. **Reflection**

* *Testing Techniques*
  + *What were the software testing techniques that you employed in this project? Describe their characteristics using specific details.*

The software testing techniques I used for project one were white box and black box testing, White box testing is defined as “A test methodology that assumes explicit and substantial knowledge of the internal structure and implementation detail of the assessment object” (White Box Testing - Glossary | CSRC, n.d.-b). Because I was developing the internal code and had very specific knowledge of the internal structure while testing for certain user requirements I implemented white box testing quite a bit. An example of the white box testing that I did was if throw errors were made if specific user requirements were not met. This required deep knowledge of the code base to implement. I also used a small amount of black box testing as well. This is defined as “A method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied to virtually every level of software testing: unit, integration, system and acceptance” (Black Box Testing - Glossary | CSRC, n.d.-b). I used black box testing to verify the desired outputs of the program without worrying about the internal structure of the code I developed. An example of this is when a user ID was generated was it placed in the string and output. If the answer was yes, the test passed. If the answer was no, the test failed.

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

One of the main testing techniques I did not use was experience based testing. Because this is my first-time implementing tests for code and programs, I do not have the experience required for this technique. “The experience-based testing technique is based on the skill and experience of the testers, experts, users etc. It is conducted in an Ad-hoc manner because proper specifications are not available to test the applications” (Faculty, 2022). Experience based testing relies on the skill and expertise of the tester. I have none of the experience necessary to perform this type of testing, so I strictly stuck to white box and black box testing.

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

The practical applications of white box testing are checking for gaps and vulnerabilities in security, broken or poorly structured paths, verifying expected output, and loop testing. White box testing can be used on many different software development projects and situations. It checks for both functional and nonfunctional user requirements as well as ensuring the code is well structured and well tested for bugs or lapses in logic. Black box testing also has many practical uses in software development projects and situations. The main being verifying input and the desired output. Black box functionality does not look at the code but focused solely on the inputs and outputs of the program. Black box tests particular inputs will have a particular output. This can be used to test the user requirements as well. If a particular input does not produce the expected output, then a potential bug has been found and the development team can work on fixing it. Experience based testing has some of the most practical uses in software development for many situations. This form of testing can be used in a time crunch where black box and white box testing may take too long. An experienced tester can implement only the tests that are specifically needed, saving the organization time and money. Experience based testing can also find more defects in the program. An experienced tester will know where to look in a program to find potential bugs. Overall experienced based testing relies on a seasoned professional utilizing their skill, intuition, and experience to find potential bugs or defects other testers may have missed.

* **Mindset**
  + *Assess the mindset that you adopted working on this project. In acting as a software tester, to what extent did you employ caution? Why was it important to appreciate the complexity and interrelationships of the code you were testing? Provide specific examples to illustrate your claims.*

The mindset that I adopted while working on this project was basically one of complete objectivity. I had zero experience with writing any form of tests prior to this course so I did my best to focus on the desired outcome and just do my best to meet the user requirements. My only focus while writing the tests and code was to ensure that all user requirements were met and tested properly. I employed caution in all areas of the development I did for both the classes and test cases. A specific example of this was after I developed a test such as a null exception and it threw an exception error I would then change the test so it would fail. I did this with all the tests I developed. I did not keep this in the end state of my project but this ensured that the tests were working correctly and not providing me with a false positive. This caution ensured that the code and test I delivered were working correctly when I was done with development. This proved to be very complicated in some respects. One complexity and interrelationship that comes to mind right away was the data in the appointment class. The relationship between the format and syntax of the date was extremely important and the entirety of that class hinged upon the successful implementation of the date not being set in the past. The testing also hinged on the correct data format being used and that took me quite a bit of trial and error before I got the relationships of the dates to work correctly based on the user requirements for both the test case and Appointment class. If I didn’t realize how important the interrelationships between the Appointment class and test cases where I may not have been able to create a working class and test case for that specific scenario.

* + *Assess the ways you tried to limit bias in your review of the code. On the software developer side, can you imagine that bias would be a concern if you were responsible for testing your own code? Provide specific examples to illustrate your claims.*

I tried to limit my bias on the review of my on code by being honest and trying to access the code like someone else had written it. I also tried to be very honest about my own shortcomings as a developer. I am still very new as a developer and my inexperience will surely be seen by someone who has a large amount experience. This is how I tried my best to eliminate my own bias. I can see how testing the code you developed could lead to having a bias. It is really hard to take criticism on something that you worked very hard on. This would be a large concern for me and whenever possible I believe that testing should be done by someone who is outside of the development team. This can take the form of a separate testing team but in many cases this is not feasible and testing must be done by the development team. So as developers we need to do our best to leave our bias at the door and look objectively at the code we develop. A specific example of this is knowing that my code has some inefficiencies and if I was working with a team on this project I would try and reach out to more experienced developers and rely on their experience to help make my code more efficient.

* + *Finally, evaluate the importance of being disciplined in your commitment to quality as a software engineering professional. Why is it important not to cut corners when it comes to writing or testing code? How do you plan to avoid technical debt as a practitioner in the field? Provide specific examples to illustrate your claims.*

Being disciplined in my commitment to quality is something that I take great pride in and will strive to maintain in all of my future projects and work. If there is a lack in quality a subpar product may be delivered to the client that could lead to loss in consumer confidence in our products and services and in some extreme cases could lead to legal action if government regulations such as security of PII (Personal Identifiable Information) are not met. This means that cutting corners should be avoided at all costs. There needs to be a balance between development and testing deadlines and the quality of all projects. In my opinion missing a deadline and avoiding cutting corners will lead to better delivered products and projects. This also means having honest communication and dialogue with the customer is extremely important. Delivering an unfinished or subpar product could lead to worse repercussions than missing a deadline. I will avoid technical debt by doing everything I can to avoid cutting corners and delivering unfinished products. I will also maintain an unbiased approach when it comes to projects I develop and work on. Also when I am unsure of how to implement a certain test or feature I will reach out to someone with more experience that’s may be able to help me. This will avoid mistakes that I may make just because I was unwilling to reach out for help. With all these strategies combined I believe I will be able to mitigate as much technical debt as possible.

**Citations**

*black box testing - Glossary | CSRC*. (n.d.-b).

https://csrc.nist.gov/glossary/term/black\_box\_testing

Faculty, P. S. (2022, August 1). *Experience based testing - Software testing technique*. H2kinfosys Blog. https://www.h2kinfosys.com/blog/experience-based-testing-2/

*White Box Testing - Glossary | CSRC*. (n.d.-b).

https://csrc.nist.gov/glossary/term/white\_box\_testing