Reflections report

Drew Shepard

12/10/2022

For my approach toward the software requirements, I tried to make sure that every requirement was met. I started with the first requirement and added on as I went down the list of requirements making sure that each requirement was working with the other so that the program worked as intended. For example, the contact class wanted a first name, last name, phone number, and address. I made sure that all of these met the character requirements and that they were not null.

I knew that my Junit tests were effective because I made sure that they met the requirement of 80% and then tried to improve the code to add as much coverage as possible. Since the coverage was 80% or greater, I knew that the code was visible to the rest of the program and that it would run the intended way.

I ensured that my code was technically sound by writing the ability to update names in my code using the following code.

@Test

void testContactServiceUpdateFirstName()

System.out.println(newContactService.contactList.get(0).getContactId());

System.out.println(newContactService.contactList.get(0).getFirstName());

newContactService.updateFirstName("1000000001", "Aaron");

System.out.println(newContactService.contactList.get(0).getFirstName());

assertTrue(newContactService.contactList.get(0).getFirstName().equals("Aaron"));

}

This ensured that the contact class's first name would update when the user wished to update only the first name.

The code was efficient because the code would print the first name that was originally set, then update the name with the choice that the user provided, and then print the new name to ensure that the name was updated in the system.

@Test

void testContactServiceUpdateLastName() {

System.out.println(newContactService.contactList.get(0).getContactId());

System.out.println(newContactService.contactList.get(0).getLastName());

newContactService.updateLastName("1000000001", "Hemsworth");

System.out.println(newContactService.contactList.get(0).getLastName());

assertTrue(newContactService.contactList.get(0).getLastName().equals("Hemsworth"));

}

From this code, you can see that the system printed the ID number to ensure that the user had the correct ID in the list. It then printed the name that was connected to the ID to ensure that the correct name was being selected to be updated. The system then updated the name with the user’s choice of name and then printed the name to ensure that the program was working as intended.

The software testing technique that I employed in my project was Dynamic Testing. Dynamic testing requires the execution of code. This gives the QA analysts the ability to see how the software will perform in a real-world environment. The main goal of dynamic testing is to ensure that the code meets the requirements. The four steps of dynamic testing are unit testing, integration testing, system testing, and acceptance testing.

Unit testing is when the code is broken down into small groups and tested individually. This was accomplished by breaking down the classes (Contact, Task, Appointment) into the smallest groups.

Integration testing is when the code is assembled into groups and tested again. Integration testing is used when we want to assemble the code and make sure that they are talking to each other.

System testing is when the code is fully assembled and tested to make sure that the code is meeting the requirements.

Acceptance testing is the last stage and is used to test the code against the requirements again to make sure that no errors made it through the other stages of testing.

I did not use Black Box Testing. Black box testing is when the tester does not have any knowledge of the software design.

I also did not use Regression Testing on this project. In regression testing, old tests will be run on new builds to make sure that the software still works as it is intended to.

For dynamic testing, the practical use is to use it on smaller projects. For a bigger project, I would recommend that static testing be used as well. This is when tests are performed on the code before it is ever written into an IDE letting QA analysts see the code before so they can catch any errors in the code. This is an extra step, but it will minimize problems later so code does not have to be rewritten or corrected. This helps minimize the cost of the project.

When working on the project I adopted a very cautious mindset to make sure that the code met the requirements of the client. I did not want the client to accept my code just to have it turn out flawed. This may have resulted in the client not wanting to use the company in future projects.

Appreciating the complexity and interrelationships of the code while testing was very important because if this was not done then I would not have thought about how the code was going to work together. Thinking about how the code works together is important because you need to be thinking one step ahead so you can minimize any problems later.

I tried to limit bias in the review of my code by making it easier to read. I thought about the end user and how they would be interacting with the program as well. I did not want the code or the program to be understandable to just me. This was code that many people would be using, and it needed to be easy to understand for the end user to enjoy using it.

The importance of being disciplined should be very high on the list of all software engineers. It is important to stay disciplined because that is how well-written code is made. Not cutting corners while writing and testing code can make it easier for the future and help clients decide that they want to come back to your company because they know that their code will be handled the correct way so they can release programs that will perform the way they want them to. To avoid technical debt as a practitioner I will take my time to make sure that the code is written well so I will not have to rework any problems later. Performing many tests on the code and making sure that they meet all the requirements of the client. If a client asks for a certain thing, I would make sure that the dev team understands them so the team can work towards writing the code to meet the requirements together.

Citations

Boog, J. (NA) 9 Types Of Software Testing In Software Engineering [9 Types Of Software Testing In Software Engineering - The QA Lead](https://theqalead.com/test-management/types-of-software-testing/)