Craig A Nelson

Department of Computer Science, Southern New Hampshire University

CS-320-T2648 Software Test Automation & QA 21EW2

Professor Kalysa Wilson

December 12, 2021

To what extent was your approach aligned to the software requirements? Support your claims with specific evidence.

The testing went hand in hand with the software requirements. For example, I was given instructions in Module 3 software requirements.:

Contact Class Requirements

The contact object shall have a required unique contact ID string that cannot be longer than 10 characters. The contact ID shall not be null and shall not be updatable.

I handled that requirement first in the Contact.java file:

![Graphical user interface, text

Description automatically generated]()

Contact Service Requirements

Further Instructions were to :

The contact service shall be able to update contact fields per contact ID. The following fields are updatable:

• firstName

• lastName

• Number

• Address

The updateable portion was handled like this:

![Text

Description automatically generated]()

Also, for the update testing in a Junit for ContactServiceTest.java We tested for Update Pass/Fail:

![Graphical user interface

Description automatically generated]()

All other test requirements for inputs or fields were handled in a similar fashion. Write the code with the requirements and then test the code. This was done as recommended in small chunks and early in the process. Ideally right after the code was written.

How did you ensure that your code was technically sound? Cite specific lines of code from your tests to illustrate.

How did you ensure that your code was efficient? Cite specific lines of code from your tests to illustrate.

My code was written according to best practices. I wrote small sections and then tested as I went to ensure that my code was technically sound. My code will be reusable for several other projects in the future. I wrote my code to be efficient by using a minimalist design, indented so my code is easy to read. Lastly, my code was commented so that any other programmer in the future will understand what is going on in the code. Two Examples Provided Below:

Some ADD Code from CONTACTServiceTest.java

![Graphical user interface, text

Description automatically generated]()

Second Example: Some DELETE Code from TASKServiceTest.java

![Text

Description automatically generated]()

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?

For Project One the requirements were to achieve 80 percent or better. I had met this mark and surpassed it with 90.9 percent. Below is a snapshot from a run In Coverage as a Junit test for Project One.

Coverage As Test for Project One:

![Graphical user interface

Description automatically generated]()

Describe your experience writing the JUnit tests.

Throughout this course, we have learned many different testing methods and dabbled in writing Junit tests for different Classes in our assigned project. Yet it seems we have barely scratched the surface of testing as a professional tester. I have much more to learn. I am glad to have the opportunity of writing Junit tests for a small, fitting exercise.

Reflection- Testing Techniques

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

We wrote Junit testing a great example of unit testing and regression testing. “Unit testing is a form of white-box testing where the tester will choose a class or a method and verify the functionality and analyze the behavior to make sure it is congruent to what is expected. So, what is a unit? It is defined by the tester, but usually the smallest piece of a class or method. This is chosen for good reason. If we choose a large chunk of code that has interaction between different pieces of code. We may get a failure but not be sure of why it failed. Regression testing and unit testing are great compliments to one another. Regression testing is done if there are any changes or updates to the code. It makes sure that the change did not break the code in any fashion. “JUnit is a Java unit testing framework that’s one of the best test methods for regression testing.” (Parasoft, 2021)

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

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

White Box vs. Black Box, Functional vs. Non-Functional, Static vs. Dynamic, Manual vs. Automated. There are so many nuances to testing. Sometimes these definitions overlap or can be confusing to categorize. The takeaway though, is let’s look at black-box testing. We were unable to use this due to the definition below because we were privy to the code and design.

* **“Black Box Testing** - Black Box Testing is a method of testing in which the internal structure/ code/design is **NOT** known to the tester. The main aim of this testing is to verify the functionality of the system under test and this type of testing requires to execution of the complete test suite and is mainly performed by the Testers, and there is no need for any programming knowledge.” (Hamilton, a, 2021)

**“Black Box Testing –** Black box can be further broken down into two types the functional and non-functional.

**Functional Testing**

In functional testing features are created are according to specifications. The QA team performs the use test cases, in the functional testing phase, the system is tested by providing input, verifying the output, and comparing the actual results with the expected results. A few Different types Of functional testing mentioned here:

* **Integration Testing**
* **System Testing**
* **Acceptance Testing**

**Non-Functional Testing**

Non-Functional testing is a testing technique that does not focus on functional aspects and mainly concentrates on the nonfunctional attributes of the system such as memory leaks, performance or robustness of the system. Non-Functional testing is performed at all test levels.

There are many Non-Functional Testing Techniques out of which the most important are:

* **Performance Testing**
* **Recovery Testing**
* **Compatibility Testing**
* **Security testing**
* **Usability testing** “(Hamilton b, 2021)

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 of a tester is to realize the importance of the work that is being conducted. Throughout this course, we researched and read many examples of software rollouts that were catastrophic and cost astronomical amounts of revenue to repair. In worse-case scenarios in the health field, there was the loss of life. *Examples listed further down paper.*

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.

Although, the presence of bias was inherent in the review processes throughout this project. I was reminding myself of this fact continuously. I work for a small business where I write code and self-test it for the last ten-plus years. I am well aware of the danger of bias and it is visible after looking over the code after a span of time.

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.

Answering the above questions and reiterating the importance of software testing, no one, especially me, would want to be responsible for any of the catastrophes of the small list below. One should not be cavalier about the responsibility of the software tester. Working for companies in the past it is the duty of the tester to learn the protocols and to follow those protocols that have been written to mitigate any catastrophic results.

“What are costly software errors?

A costly software error can come in many different shapes and sizes, and the cost isn’t always monetary. Software is in almost everything we do, and the consequences when things go wrong can be devastating. From costing multiple billions of dollars to causing real human lives to be lost, broken software has the potential to have unthinkable consequences.

* NASA’s Mars Climate Orbiter
* Ariane 5 Flight 501
* EDS Child Support System
* Soviet Gas Pipeline Explosion
* Bitcoin Hack, Mt. Gox
* Heathrow Terminal 5 Opening
* The Mariner 1 Spacecraft
* The Morris Worm
* Patriot Missile Error
* Pentium FDIV bug
* Knight’s $440 Million Error
* The NOA-19 satellite”

(|, 2021)

One last note we are all human individually and as a group and are prone to mistakes. I think we should be cognizant of that fact.

“Quality is not an act, it is a habit.”— Aristotle (Palamarchuk et al., 2020)

**References**

Hamilton, T. (2021a, October 8). Static testing vs Dynamic Testing: What's the difference? Guru99. Retrieved November 7, 2021, from

<https://www.guru99.com/static-dynamic-testing.html>

Hamilton, T. (2021, October 8). What is dynamic testing? types, Techniques &amp; Example. Guru99. Retrieved November 7, 2021, from

<https://www.guru99.com/dynamic-testing.html>

Junit tutorial with examples: Setting up, writing, and running Java Unit tests. Parasoft. (2021, June 7). Retrieved December 12, 2021, from

<https://www.parasoft.com/blog/junit-tutorial-setting-up-writing-and-running-java-unit-> tests/.

|, N. H. (2021, February 5). 11 of the most costly software errors in history. Raygun Blog. Retrieved December 12, 2021, from

<https://raygun.com/blog/costly-software-errors-history/>.

Palamarchuk, S., Palamarchuk, S., Kaner, D. C., Rutan, B., Molyneaux, I., Weinberg, J., Booch, G., &amp; Yip, J. (2020, July 24). The ultimate list of 100 software testing quotes: Abstracta. Abstracta Software Testing Services. Retrieved December 12, 2021, from https://abstracta.us/blog/tools/ultimate-list-100-software-testing-quotes/.