**Project Two**

Felicia Schenkelberg

Southern New Hampshire University

CS 320: Software Testing, Automation, and Quality Assurance

Professor Moscon

18 April 2021

# Introduction

As a software engineer for Grand Strand Systems, I completed a project in which I developed a mobile application for a client and delivered the contact, task, and appointment services. I now will make a summary and reflections report which summarizes my unit testing approach, experience writing the JUnit tests, and overall quality of my JUnit tests. I will also point out the testing techniques/mentality I adopted while working on this project.

# Summary: Alignment to Requirements

Testing is an activity used to reduce risks and improve quality by detecting defects. During this SDLC testing phase, I reported, tracked, and fixed product defects until the product meet the quality standards specified in the SRS (Hambling et al., 2015). The “fundamental test process consists of five parts that encompass all aspects of testing:

1. Planning and control.
2. Analysis and design.
3. Implementation and execution.
4. Evaluating exit criteria and reporting.
5. Test closure activities” (Hambling et al., 2015).

The testing phase is critical to the success of SDLC as it reveals product defects. Testing activities should begin as soon as possible and involve all stages of the SDLC. This is because errors are much cheaper to fix than defects (or failures). The software testing system includes both static and dynamic testing. “Static testing is the term used for testing where the code is not exercised… [and] involves techniques such as reviews... Dynamic testing is the kind that exercises the program under test with some test data” (Hambling et al., 2015).

Dynamic and performance testing both aid in the alignment of my service requirements. These specific services differed from feature to feature. However, during performance testing, the services were tested and proved my code aligned to the software requirements.

Task Class Requirement

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

Task Class Code

*package taskClass;*

*public class Task {*

*private String taskID;*

*int maxLength = 10;*

*public Task(String taskID, String Name, String Description) {*

*if (taskID.length() > maxLength && taskID != null) {*

*this.taskID = taskID;*

*}*

*}*

*}*

## Summary: Effective Tests

While static testing is the term used for testing the software without execution. Dynamic testing, on the other hand, is the term used for testing the software by execution (Hambling, 2015). The main difference between static and dynamic testing is the use of some test data. Static testing does not use test data as it relies on methods non-reliant on software execution. Dynamic testing, however, does exercise the software program with some test data. It is important to remember that the discipline of software testing “encompasses both static and dynamic testing” (Hambling, 2015).

Static techniques are important because they can find errors and defects before executing the code. Dynamic testing is important as it detects defects when executing the code. Both techniques are done to identify errors and reduce costs. This is because errors are much cheaper to correct than defects or failures (Hambling, 2015). These techniques make testing much more effective. Throughout this process, I determined that “Good tests emphasize variety and complexity. So, [I]

1. Use[d] the same test case patterns in different ways and in different contexts...
2. [Tried] to combine test cases in random, unusual, strange, or weird ways” (Chopra, 2018).

## Summary: Technically Sound Code

I ensured that my code was technically sound by also undergoing a process of carefully and methodically reviewing the software design, architecture, or code for bugs without execution. This structural analysis is sometimes referred to as static testing. “Static Testing is a type of testing in which the program source code is tested without running it” (Chopra, 2018). In doing this, I needed to only examine and review the code to determine if the code worked according to the functional requirements and was written in accordance with the design developed earlier in the project life cycle (Chopra, 2018).

## Summary: Efficient Code

I ensured that my code was efficient by following a fundamental principle. The smallest programming unit should be tested first and then expanded to other parts of the system (Chopra, 2018). An example of this was developing and testing the Task Class before developing and testing the Task Service.

## Reflection: Techniques Employed

The purpose of testing is to find undiscovered errors. Code reviews, inspections, and walkthroughs are three common forms of static testing. Alternatively, Dynamic testing occurs when the code is ready. Three common forms of dynamic testing are black box, white box, and gray box testing.

“The term Black-Box refers to the software which is treated as a black-box. By treating it as a black box… the system or source code is not checked at all… White-box testing is a way of testing the external functionality of the code by examining and testing the program code that realizes the external functionality… White-box testing is used to test the program code, code structure, and the internal design flow” (Chopra, 2018). Gray-Box testing is “a combination of white-box and black-box techniques…” (Chopra, 2018).

The software testing technique I used, for each of these services, was a form of white box testing. This is because my testing focused on the internal structure. While performing unit tests, I was able to select inputs to explore specific paths and determine the appropriate output. The purpose of this software testing technique was to examine the individual components, verify the functionality, and ensure that the behavior was as expected.

## Reflection: Other Techniques

The software testing technique that I did not use for the milestones was the Black-Box (or Functional) Testing Technique. This is because I frequently checked my source code.

## Reflection: Uses and Implications of Techniques

The practical uses for White-Box (or Structural) Testing are isolated, focused testing. In conducting individual, targeted tests, developers can not only ensure the behavior of individual methods but also check for invalid input. It is also helpful to deploy powerful unit testing tools to save valuable time for developers.

The practical uses for Black-Box Testing are verification and validation. In conducting tests, developers can not only verify and validate the entire software development life cycle but also evaluate the product at various stages. It is also helpful to see the program from the customer’s viewpoint.

The practical use for Gray-Box testing is the incorporation of elements from both black-box and white-box testing. In conducting tests, developers consider the outcome on the user end, system-specific technical knowledge, and the operating environment. It is also helpful to evaluate the application design in the context of the interoperability of system components.

## Reflection: Caution

If I were to assess the mindset that I adopted working on this project, I would first have to address the main step I underwent to avoid mistakes, fixing defects. A few were easily managed such as typos, missing features, and so on. However, I found it important to use caution and appreciate the complexity of the code I was testing.

## Reflection: Bias

I tried to limit bias by employing a testing technique called random testing. In doing this, I was able to not only randomly pick test case values but also execute stress tests. “Tests that stress the system with random inputs (like the number of users, size of data, etc.) at random instances and random magnitude are selected and executed as part of stress testing” (Chopra, 2018). Bias is a very real concern for testing computer systems as it creates unfair outcomes. One example is privileging one arbitrary group of users over another.

## Reflection: Discipline

The importance of discipline is undeniable. Management often needs to track cost, schedules, and/or even functionality. As such, it is important not to cut corners when it comes to writing or testing code. The concept of discipline also fits in with Quality Standards. This is because projects are then repeatable. I plan to avoid technical debt by avoiding additional rework caused by selecting an easier solution rather than finding a better approach.

References

Chopra, R. (2018). Software Quality Assurance: A Self-Teaching Introduction. Mercury Learning & Information.

Hambling, Brian Morgan, Peter Samaroo, Angelina Thompson, Geoff Williams, Peter. (2015). Software Testing - An ISTQB-BCS Certified Tester Foundation Guide (3rd Edition) - 1.7 Fundamental Test Process. BCS The Chartered Institute for IT.