CS-320-T4208 Software Test Automation & QA

Module Five Journal – Software Testing Techniques

Computer Science Department

Southern New Hampshire University

Eric Farkas

4/2/2023

Modules three, four and five focused heavily on unit testing using JUnit. This type of testing is generally classified as white box testing. This is because we were focused on the internal functionality of the code and adherence to rigid specifications. Additionally, development and testing performed by the same person is considered white box testing. The approach is loosely defined as incremental integration testing even though the approach does not test integration. This approach allowed me to focus on the requirements and ensure adequate coverage before moving to the next field within each object. This approach seems very efficient for small developments carried out by a single developer or a close team but I can see how this approach would become problematic in a larger development team. There were specific questions defined within the assignment that are addressed individually below.

**What were the software testing techniques that you employed for each of the milestones?**

The approach and techniques I used changed over the course of the assignments. I performed the code development for modules three and four first and then created the test code. Module five was a more hybrid approach. In all cases, I reviewed the specifications and created documentation to reference during development and testing. The documentation captured specifications and names to ensure consistency throughout the code and support best practices. It also captured maximum and minimum field length and input validation strategies to ensure the data entered was correct and formatted. The built-in functionality of the JUnit framework made the testing code quite easy as the testing parameters are simply inserted based on defined conditions separated into sections of testing code. The coverage utility counts the conditions and shows the count and percentage of conditions the testing addresses. This creates an excellent cross-reference between the design documents and the test results. The input validation was further tested with try & catch statements in the code and user feedback was added to any exception to help the user provide valid input.

**What are the other software testing techniques that you did not use for the milestones?**

The testing for these milestones focused heavily on function. Many non-functional elements such as performance, security, scalability and interoperability were out of scope. No black box testing was performed during these milestones as the code and testing were performed by a single person. There may be some opportunity to expand the scope to include some of these missed techniques before the class concludes. Additional requirements will be necessary to improve the documentation and determine the testing appropriate to support these new requirements.

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

The white box testing performed makes a lot of sense for the developer to perform. It raises the product quality as the developer has to plan more thoroughly to handle improper data. I have little experience but it seems this strategy works well on simple product development with small teams. It would be cumbersome and time-consuming with larger developments and difficult to normalize in large teams. The fundamentals are still valuable and should be used as best practice when the scope is manageable. The input validation tests are always necessary. Too many documented vulnerabilities and attack methodologies exist to allow unrestricted data fields on public-facing systems. My strategy should significantly reduce quality and user testing cycles and allow the product to release to production faster.