CSE 565 Portfolio Submission

Member Name: Gautham Vijayaraj
ASU ID: 1229599464
Date: 12/4/2024

Introduction

This report summarizes the key learnings and outcomes from five comprehensive assignments completed in this course (CSE 565: Software Verification and Validation). Each assignment focused on specific aspects of software verification and validation, leveraging tools, methodologies, and testing paradigms. The assignments strengthened my understanding of concepts like unit testing, combinatorial testing, code coverage analysis, endurance testing, and reliability prediction. This portfolio reflects the application of theoretical knowledge to practical scenarios, equipping me with essential skills for software quality assurance.

Assignment 1: Generative AI for Unit Testing

Introduction

This assignment explored the use of **Diffblue Cover**, an AI-driven tool for generating unit tests, and compared its functionality with the manual approach of writing test cases in **JUnit**.

Summary of Results

- Diffblue Cover successfully automated the generation of unit test cases for a Quick Sort implementation, providing comprehensive coverage for edge cases, boundary values, and normal scenarios.
- Generated test cases enhanced the test suite efficiency, covering multiple execution paths and reducing manual effort.
- Comparative analysis showed that while Diffblue Cover automates baseline testing, JUnit allows for manual fine-tuning of complex, domain-specific test cases.

New Skills and Knowledge

- Learned to integrate AI tools into the development lifecycle for test automation.
- Understood the complementarity of automated and manual testing approaches.
- Gained proficiency in identifying scenarios where AI can accelerate development without compromising test accuracy.

Assignment 2: Design of Experiments and Pairwise Testing

Introduction

This assignment employed the **PICT (Pairwise Independent Combinatorial Testing Tool)** to optimize test case generation for an e-commerce platform with multiple variable combinations.

Summary of Results

- **PICT** reduced an exhaustive testing set to 20 pairwise combinations, ensuring comprehensive interaction coverage while minimizing effort.
- Key features tested included combinations of authentication methods, payment systems, shipping options, device types, and user account types.
- Generated compact test cases were effective in exposing potential interactions and conflicts between system components.

New Skills and Knowledge

- Mastered the application of **Design of Experiments (DOE)** to software testing.
- Learned to leverage pairwise testing to optimize test case design without sacrificing coverage.
- Acquired hands-on experience using PICT on macOS through Mono integration.

Assignment 3: Code Coverage Analysis and Static Source Code Analysis

Introduction

This assignment combined **JaCoCo** for decision code coverage and **SonarLint** for static source code analysis to evaluate a Java-based Quicksort implementation.

Summary of Results

- **JaCoCo** identified untested branches and logical paths, enabling targeted test enhancements to achieve full decision coverage.
- **SonarLint** highlighted issues like unused variables, dead code, and reassignment errors, ensuring adherence to coding best practices.
- Combined, these tools provided a comprehensive overview of both runtime behavior and static code quality.

New Skills and Knowledge

- Developed expertise in integrating JaCoCo into a Maven project to visualize and improve test coverage.
- Gained proficiency in real-time static code analysis with SonarLint, enhancing code maintainability.
- Understood the synergy between dynamic and static analysis tools for robust software testing.

Assignment 4: Endurance Testing

Introduction

This assignment focused on endurance testing, analyzing the prolonged stability of systems like e-commerce platforms and banking applications under sustained load.

Summary of Results

- Used tools like **Apache JMeter** and **LoadRunner** to simulate real-world conditions, such as peak user traffic and continuous transaction processing.

- Identified critical issues, including memory leaks, database connection failures, and response time degradation, in long-duration tests.
- Proposed actionable strategies to ensure system reliability under extended operational loads.

New Skills and Knowledge

- Acquired hands-on experience with endurance testing tools for prolonged stress simulations.
- Learned to identify and address system bottlenecks that emerge under continuous usage.
- Gained insight into designing realistic load scenarios for stability testing.

Assignment 5: Reliability Prediction Approaches

Introduction

This assignment compared traditional reliability models like the **Musa-Okumoto model** with modern, data-driven approaches such as machine learning.

Summary of Results

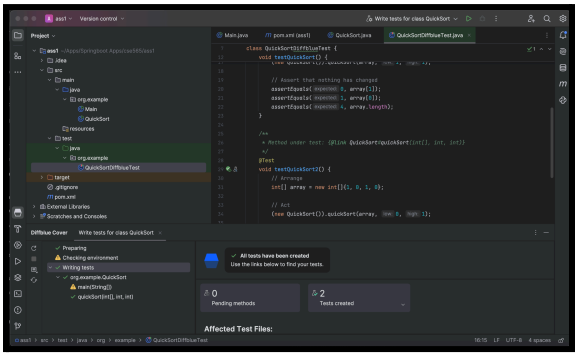
- **Musa-Okumoto Model** proved effective in scenarios with well-documented failure data but lacked adaptability to dynamic systems.
- Machine learning-based methods demonstrated flexibility, leveraging diverse datasets like code metrics and usage logs for predictions.
- Highlighted the trade-offs between the data dependency of traditional models and the scalability of machine learning approaches.

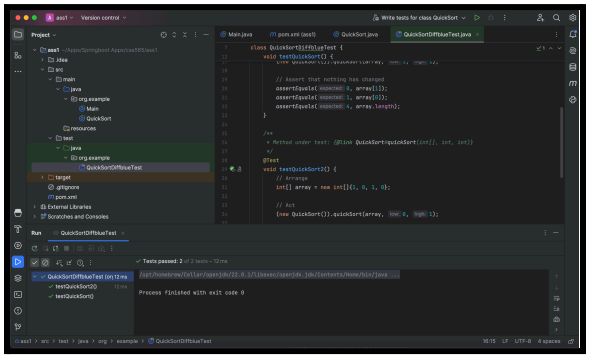
New Skills and Knowledge

- Gained an understanding of the **logarithmic Poisson execution time model** for reliability prediction.
- Learned to apply machine learning algorithms for dynamic and heterogeneous software systems.
- Developed the ability to evaluate predictive models based on context, data availability, and system requirements.

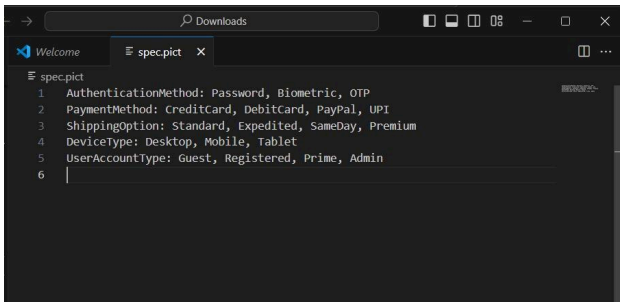
Output Screenshots for the Assignments (Only for the practical assignments)

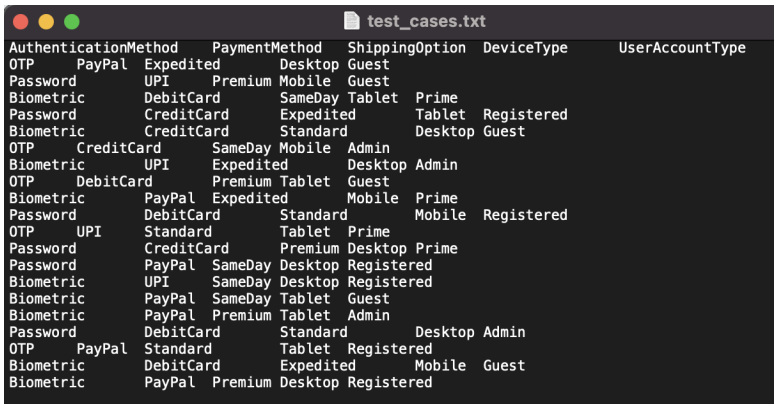
1. **Assignment 1:** Test cases generated by Diffblue cover and the test cases passing.



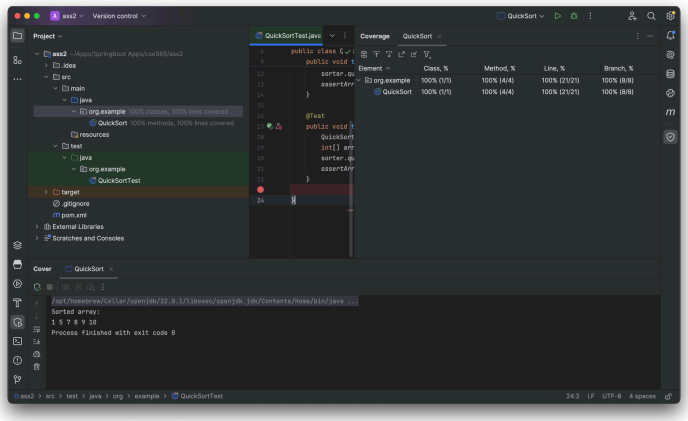


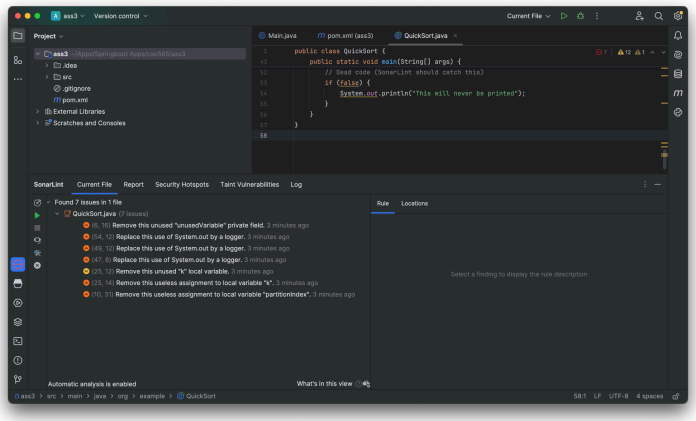
2. **Assignment 2:** The pict file with the specifications & factors and the output test cases generated.





3. **Assignment 3:** Analysis provided by JaCoCo (coverage reports) and SonarLint (quality reports).





Conclusion

These assignments collectively enhanced my understanding of software testing paradigms, instilling both theoretical knowledge and practical skills. From leveraging AI for automated testing to optimizing reliability predictions, I gained a holistic view of ensuring software quality. Tools like Diffblue Cover, PICT, JaCoCo, SonarLint, JMeter, and LoadRunner became instrumental in developing a rigorous and efficient testing approach. This learning experience prepares me to tackle complex testing challenges in real-world projects.

References

1. Diffblue Cover Documentation - Diffblue Documentation
2. ChatGPT Prompt for assignment 1 - ChatGPT Prompt 1
3. Design of Experiments (DOE) Glossary - What Is Design of Experiments (DOE)? | ASQ.
4. PICT Documentation: PICT Data Source - Windows drivers | Microsoft Learn
5. PICT Tool GitHub Repository - Pairwise Independent Combinatorial Tool
6. ChatGPT Prompt for assignment 2 - ChatGPT Prompt 2
7. JaCoCo GitHub Repository - Java Code Coverage Library
8. JaCoCo Installation Steps - JaCoCo Java Code Coverage Library
9. SonarLint Documentation - SonarLint Plugin for JetBrains IDEs
10. Gemini Chat Prompt for assignment 3 - Gemini Prompt 1
11. ChatGPT Prompt for assignment 3 - ChatGPT Prompt 3
12. ChatGPT Prompt 2 for assignment 3 - ChatGPT Prompt 4
13. Endurance Testing Definition - Endurance Testing | Soak Testing - ArtOfTesting
14. Endurance Testing Guide - Endurance Testing Guide: How To Perform
15. QA Testing Instights - Endurance Testing - A Complete Guide by QASource in 2024
16. Endurance Testing for System Stability - Endurance Testing and Its Importance
17. LoadRunner Documentation - LoadRunner Tutorial
18. Apache JMeter Documentation - Apache JMeter
19. ChatGPT Prompt for assignment 4 - ChatGPT Prompt 5
20. Musa, J. D. (1980). *Software Reliability Engineering: More Reliable Software Faster and Cheaper*. McGraw-Hill.
21. Morales, R., Gao, S., & Pattipati, K. R. (2021). "A machine learning approach for software reliability prediction." *Journal of Systems and Software, 176*.
22. Zhang, X., Li, Y., & Wang, S. (2020). "Machine learning methods for software reliability prediction." *IEEE Transactions on Reliability, 69(3)*.
23. Singh, A., & Singh, M. P. (2014). Software reliability growth modeling using machine learning techniques: A survey. International Journal of Computer Applications, 97(21), 1-7.
24. Musa-Okumoto Logarithmic Model Definition : GeeksforGeeks Link
25. ChatGPT Prompt for assignment 5 - ChatGPT Prompt 6