Testing Report

Testing Report | Milestone 3

Cameron Seppi, Vadim Pidoshva, Jessica Adams, Trajan Clark
TEAM 10, CS-4230, Dr. Jingpeng Tang

Table of Contents

1. Introduction

Overview of Testing Objectives

The main goal of our testing efforts was to evaluate both our own program and a peer group's program. We sought to ensure that both programs functioned correctly, met their requirements, and could handle a variety of valid and invalid inputs. The testing aimed to uncover bugs and flaws in functionality, as well as assess the programs' reliability and stability.

Importance of Methodical Testing

Testing was crucial to ensure both programs would meet user expectations and perform reliably under real-world conditions. A systematic, methodical approach was necessary to ensure we covered a wide range of scenarios, including edge cases, and potential user errors.

---

2. Testing Methods Selected

Methods Chosen and Their Rationale

We selected **Equivalence Class Testing** and **Boundary Value Testing** for their effectiveness in validating input constraints and ensuring the programs handled a range of scenarios. We also used **Code Reviews** for higher code quality.

Evaluation of Testing Methods Chosen

- **Equivalence Class Testing**: It allowed us to efficiently group inputs into valid and invalid categories, this method helped ensure all possible input scenarios were covered.
- **Boundary Value Testing**: Critical for testing how the programs handled edge cases, particularly when dealing with ranges of inputs like employee counts or the number of loans.
- **Code Review:** In addition to the above methods, we implemented a Code Review process to ensure code quality and functionality. Code changes were submitted via GitHub Pull Requests (PRs).

Alternative Methods Considered and Rejected

- **State Transition Testing**: We initially considered using state transition; however, given the complexity of the programs we determined that it would be difficult to plan and implement state transition tests effectively.

3. Deviations from the Original Test Plan

Testing Plan Detours

In addition to the original test plan, we performed the following testing activities:

1. **Reliability Testing**: We wanted to ensure both programs could run for extended periods without crashing. We simulated long-running scenarios to check their stability over time.
2. **Validity Testing**: We tested how the programs handled incorrect inputs (e.g., negative numbers, invalid characters) to see how they would react and recover. This was particularly important as we could only test with limited sample data.

Testing Blockers

During the testing process for the other team's program, we discovered that the program was incomplete and missing functionality. Due to this, we were unable to perform a number of our originally planned tests.

---

4. Criteria for Ending Testing

Decision-Making Process for Ending Testing

We decided to stop testing once the following criteria were met:

1. Most critical tests passed.
2. No major bugs remained unidentified.
3. The program could handle common use cases and was stable for long periods.

Factors Considered

- **Coverage**: We ensured that all equivalence classes and boundary values were tested.
- **Time Constraints**: Given the time limitations, we prioritized the most essential tests and identified critical bugs to ensure that the programs were thoroughly tested.

---

5. Comparative Evaluation of Programs Tested

Testing Our Program: Dysfunctional Organization System

Testing our own program was straightforward because we were familiar with its design. However, we had to be vigilant about missing functionality and incomplete features, which affected the depth of our tests.

- ### Effectiveness of Testing Methods

  Equivalence class testing identified logical gaps in rule enforcement, such as exceeding role limits. Boundary value testing proved essential in discovering critical errors when maximum constraints were violated. Code reviews helped reduce trivial errors but missed deeper logic flaws.

- ### Evaluation of Testing Effectiveness

  The selected approaches covered a wide range of functionality but revealed the need for additional automated regression testing. Some bugs, such as cascading role promotions, emerged only in complex scenarios not explicitly planned for.

- ### Criteria for Stopping Testing

  Testing concluded once all critical functionalities were covered and 100% of planned test cases passed with no critical failures. Risks of uncovered edge cases remained minimal based on priority and frequency.

Testing the Peer Program: WeCheatEm Bank System

Testing the peer's program posed challenges, especially since much of its functionality was not yet fully implemented. Still, we focused on testing the areas that were operational, and the process helped identify areas that required further development.

- ### Effectiveness of Testing Methods

  Boundary value testing effectively exposed critical bugs, including incorrect interest calculations for extreme loan amounts. Statement coverage testing validated the thoroughness of execution paths but required refinement for completeness.

- ### Evaluation of Testing Effectiveness

The approaches detected several bugs, such as interest miscalculations and failure to enforce input limits. However, gaps remained in identifying UI-level issues and user experience flaws.

- Criteria for Stopping Testing

  Testing ceased when critical features were tested to the best of our limits and all bugs critical to functionality were identified. The team decided based on time constraints and project scope.

Lessons Learned

1. **Flexibility in Testing Plans:**
   Adjusting strategies mid-testing revealed issues outside the initial scope, enhancing overall quality.
2. **Importance of Automated Testing:**
   Regression testing could have saved significant manual effort and reduced risk of reintroducing bugs.
3. **Communication and Documentation:**
   Clear documentation of test cases and results significantly eased the testing and debugging process. Testing both programs highlighted the importance of communication and clear project requirements.

---

6. Reflections on Testing Principles

Lessons Learned about Testing Principles

- Comprehensive coverage is essential, even when testing incomplete systems.
- It's important to adjust testing strategies based on the stage of development, balancing thoroughness with available resources.

Understanding of Tradeoff in Testing Approaches

We learned that testing early versions of a program requires flexibility. While Equivalence Class and Boundary Value Testing were effective for input validation, we had to supplement them with other testing approaches (like code reviews and reliability testing) to identify and resolve issues caused by incomplete functionality.

---

7. Conclusion

Summary of Testing Experiences

Our testing process successfully uncovered critical bugs in both programs, validated the handling of valid and invalid inputs, and ensured some degree of reliability for both systems. Even though one of the programs was not fully implemented, the tests we performed helped clarify the status of key features and identified areas requiring further attention.

Final Thoughts on Effective Testing

Testing both our program and the peer group’s program emphasized the importance of methodical approaches, communication, and adapting testing strategies based on the system's development stage. While our testing efforts were effective given the constraints, future projects should incorporate more automated testing to improve efficiency and coverage.