# SFT221 SCRUM Report and Reflections

This report should be completed in the class and submitted at the end of class. Late submissions cannot be accepted without prior approval of the instructor.

**GROUP: 1**

**Members Present**:

|  |  |
| --- | --- |
| 1. Hoi Kit Cheung | 4. Sau Ching Yuki Wong |
| 2. Gyeongrok oh | 5. Yonghun Won |
| 3. Lap Chi Wong | 6. Pui Wai Tse |

## Milestone 5 Tasks

In this milestone, you should write, implement, and execute integration tests. Integration tests test how multiple functions work together to complete a task. Depending on what is being tested, you might be able to write unit tests to do the testing and automatically compare the results. In other cases, you might need to manually check the output to check it. This will all be stated in the tests where it discusses how they should be run.

As you update the function-test matrix, you will need to add a very brief description for each integration test so the matrix will clearly show what the tests are testing. Acceptance tests will be tested against actual user requirements and will list all the tests for each requirement.

Acceptance tests are the final tests and are largely aimed at showing the customer that the correct output is produced for different inputs. This will largely require manual testing.

**Deliverables Due at end of Lab:**

* Completed SCRUM report and reflections

**Deliverables Due at 23:59 12 Days after Lab:**

* integration tests written and stored in repository,
* integration tests written (store in repo), executed (results in Jira and in test documents) and debugged.
* acceptance tests written and stored in repository.
* Updated function-integration-requirements-test matrix stored to the repository.

**Rubric**

|  |  |  |
| --- | --- | --- |
| Individual | Group Participation | 75% |
| Teamwork | 10% |
| SCRUM Report and reflections | 15% |
| Group | integration tests (well-designed, written and documented) | 20% |
| acceptance tests (well-designed, written and documented) | 20% |
| Test Execution (performed, results recorded, issues created) | 15% |
| Debugging (Bugs fixed, documented, Jira updated) | 5% |
| Function-test matrix updated | 5% |
| Git Usage (used properly with good structure) | 5% |
| Jira Usage (creates issues, tracks progress) | 5% |
| Meets Deadlines | 5% |
| SCRUM Report and Reflections | 20% |

**SCRUM Report**

**Summary of Tasks Completed or Delayed in the last week:**

Here you can list all of the tasks completed in the last week along with any tasks which could not be completed with a reason why they could not be completed.

|  |  |  |
| --- | --- | --- |
| **Member** | **Tasks Completed** | **Tasks Delayed/Blocked** |
| Hoi Kit Cheung | * Write and execute integration test codes (Visual Studio) for test group 1 and 3 * Prepare test report templates * Update function-test matrix / Acceptance Test * Reflection Question 1 | None |
| Gyeongrok oh | * Write integration test cases (Excel) and test codes (Visual Studio) for test group 2 * Prepare Scrum Report | None |
| Lap Chi Wong | * Identify function groups to perform meaningful integration tests for test group 1, 2 and 3 * Write the codes of the main function and perform testing * Debug for failed test cases reported | None |
| Sau Ching Yuki Wong | * Write integration test cases (Excel) for test group 1 * Reflection Question 4 | None |
| Yonghun Won | * Write integration test cases (Excel) group 3 * Reflection Question 3 | None |
| Pui Wai Tse | * Write integration test cases (Excel) group 1 * Reflection Question 2 | None |

For every task delayed or blocked, describe the reason for the delay or block, how it impacts the project and the proposed solution or workaround**.**

|  |  |
| --- | --- |
| **Delayed or Blocked Task** |  |
| **Reason for delay or block** |  |
| **Impact on Project** |  |
| **Solution or work-around** |  |
|  |  |
| **Delayed or Blocked Task** |  |
| **Reason for delay or block** |  |
| **Impact on Project** |  |
| **Solution or work-around** |  |

**Summary of Meeting:**

A summary of the main points discusses in the meeting and the outcomes of the discussions.

|  |  |  |
| --- | --- | --- |
| Topic | Discussion Summary | Outcome |
| Workload distribution | To distribute the tasks for milestone 5 to team members so that each member has a fair share of workload | The workload distribution is agreed in the meeting |
| Discussed project timeline | Defined project timeline and deadlines. Adjusted the timeline based on everyone's needs and ensured that all team members agreed on the deadlines. | Project timeline finalized with adjustments based on team consensus. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Summary of Decisions Made:**

This will include major architecture and design decisions, testing decisions, prioritization of tasks, dealing with problems encountered and other major outcomes from the meeting.

|  |  |
| --- | --- |
| Decision | Rationale |
| Testing Strategy Enhancement | It was decided to focus on improving the testing strategy to ensure all implemented functions meet the required quality standards. This decision was prompted by the identification of a failed test and the recognition that all tests should pass seamlessly to ensure reliable software. |
| Increased Documentation and Collaboration | The team agreed to enhance collaboration and communication among team members by incorporating more comments in Jira tickets. This decision arose from the feedback that greater transparency and insight into individual contributions are essential for tracking progress effectively. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Tasks Attempted During Meeting:**

Each member is assumed to participate in the SCRUM meeting and contribute to the completion of the SCRUM report and reflections. Since the SCRUM meeting will not take more than 20-30 minutes, there is lots of time left to undertake some of the actual work tasks. In the table below, each member should list what they did to complete the SCRUM report, the reflections, and 1-4 other tasks they completed during the class period. If a task could not be completed, the student should indicate why this was not possible.

|  |  |  |  |
| --- | --- | --- | --- |
| Member | Task Attempted | Time Spent | Complete? |
| Gyeongrok, Oh | Reflections, code review | 30 mins | done |
| Hoi Kit, Cheung | Reflections, code review | 30 mins | done |
| Lap Chi, Wong | Reflections, code review | 30 mins | done |
| Pui Wai, Tse | Reflections, function review | 30 mins | done |
| Sau Ching Yuki, Wong | Reflections, function review | 30 mins | done |
| Yonghun Won | Reflections, function review | 30 mins | done |

**SCRUM Tasks Selected for Next Week**:

The tasks each member has selected to pursue for this class or the next week.

|  |  |
| --- | --- |
| Group Member | Task Description |
| take turns | meeting record (scrum report) |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Major Outcomes of Meeting:**

This is where you should highlight the major accomplishments of the class.

|  |  |
| --- | --- |
| Outcome | Impact on Project |
| Successful Integration Validation | The integration test was executed to verify the seamless interaction between the newly implemented features and the existing system components. The team confirmed that the integration was successful without any major conflicts or disruptions. |
| Data Integrity Verification | The team examined the system's data integrity after the integration to confirm that data structures and values remained consistent and accurate. |
| Validation of User Flows | The integration test included validation of user workflows that involve the new features. This step ensured that the end-to-end user experience remains smooth and uninterrupted. |
|  |  |
|  |  |
|  |  |
|  |  |

**Things That Went Well in This Meeting:**

Here you can highlight things which worked well. This indicates that the way you worked on these items is working and should be continued.

|  |  |
| --- | --- |
| Topic/Work Item | Reason for Success |
| Integration test design | Relevant functions are grouped to together to perform meaningful integration tests |
| Workload distribution | Requirements of Test Case and code are fully understood by all team members to create reasonable sub-tasks |
| Project Timeline | Realistic and achievable milestones set |
| Responsibilities | Clear understanding of individual tasks and roles |
|  |  |
|  |  |
|  |  |

**Things That Did NOT go Well in This Meeting:**

This is where you can list things which did not go well in the class. You should analyze why this happened and suggest how you can improve it next time. This will lead to the goal of *continuous process improvement*.

|  |  |
| --- | --- |
| Topic/Work Item | Reason for Problem and How to do Better |
| NIL |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Reflections**:

1. At this point, you are using the GIT hook to automate testing. Have you found that any of the tests failed and prevented you from pushing your code to the repository? If so, how did you handle the situation?  
     
   Yes, I have encountered some situations where the tests failed and prevented me from pushing my code to the repository. For example, once I was working on a feature that required me to modify a function in a module that was used by other parts of the code. I ran the unit tests locally and they passed, but when I tried to push my code, the GIT hook ran the integration tests and some of them failed. This meant that my changes had broken some functionality in other modules that depended on the function I modified.

I handled the situation by first checking the error messages and logs from the failed tests. They gave me some clues about what went wrong and where to look for the problem. Then I used a debugger to step through the code and inspect the variables and outputs. I also used some print statements to see what values were being passed and returned by the function. After some debugging, I found out that I had made a mistake in the logic of the function and it was returning incorrect results for some edge cases. I fixed the bug and ran the tests again. This time they all passed and I was able to push my code successfully

1. Explain why we are automating the testing process and what the advantages of this automation are.  
     
   Automating the testing process is a way to make testing computer programs faster and more accurate. This is important because it helps us find problems in the program quickly. Imagine if we had to test the program by hand every time we make a change - that would take a lot of time and we might miss some mistakes. With automation, the computer can run tests for us, which means we can find mistakes sooner and fix them before they become big problems.

Automating have several advantages:

First, automation is faster. Computers are super-fast at running tests, much faster than humans can manually go through them. This means we can test our software thoroughly without wasting too much time.

Second, automation is more reliable. When people do tests, there can be small differences each time – maybe we forget a step or do things slightly differently. Computers, on the other hand, are very consistent. They follow the exact same steps every single time, ensuring that tests are carried out accurately and without any variation.

Third, automation enables efficient regression testing. This means that every time we make changes or improvements to our software, we can quickly and automatically re-test everything that used to work well. This helps catch any unexpected side effects of our changes, preventing new issues from creeping into our software.

1. Did you find the integration and acceptance tests more difficult to write than the black box and white box tests? If so, why were they harder to write? Did you write more white box and black box tests or more integration and acceptance tests?  
     
   That’s a good question. I think the integration and acceptance tests were more difficult to write than the black box and white box tests. There are several reasons for that Integration tests require me to test the interaction between different modules or components of the system. This means I need to have a good understanding of how each module works and what are the expected inputs and outputs. I also need to consider the possible scenarios where the modules may fail or produce unexpected results. This can be challenging, especially if the modules are complex or poorly documented. but Acceptance tests require me to test the system from the user’s perspective. This means I need to have a clear idea of what are the user’s requirements, expectations, and preferences. I also need to design realistic and representative test cases that cover the main features and functionalities of the system. This can be time-consuming, especially if the system is large or has many variations.

Black box and white box tests are easier to write because they focus on testing individual units or functions of the system. This means I only need to know the specification or implementation of the unit or function I’m testing. I can use tools like code coverage or boundary value analysis to help me generate test cases. This can be faster and simpler, especially if the units or functions are well-defined and modular.

I wrote more white box and black box tests than integration and acceptance tests. This is because white box and black box tests are more suitable for testing the lower levels of the system, such as the logic, algorithms, data structures, and interfaces. Integration and acceptance tests are more suitable for testing the higher levels of the system, such as the usability, reliability, performance, and security. Since I was mainly working on developing and debugging the lower levels of the system, I focused more on white box and black box tests.

1. Explain why it is necessary to write integration and acceptance tests given that all of the code has already passed black box and white box tests.

Integration tests focus on verifying the interactions and compatibility between different function or modules of the program. Even if all the single black box and white box test case of each function are passed, the function might not able work well with the other function. Black box or White box test case are individual and isolated in unit testing. The integration tests validate the collaboration and data flow among these functions. The integration tests can help detect issues, such as data inconsistencies, and unexpected behaviors during functions interactions, and communication failures. Integration tests provide a more comprehensive view in the software testing and the functions cohesiveness across the entire program.

Acceptance tests focus on validating whether the software meets the specified requirements and fulfills the end user’s needs, acceptance tests assess the software's alignment with business objectives and user expectations. These tests simulate real-world scenarios, user interactions, and workflows to ensure that the software delivers the intended value. Those tests also provide a final validation of the software's usability, performance, and security. Passing acceptance tests established confidence to stakeholders, including clients and users, that the software is ready for deployment and use.

In conclusion, integration and acceptance tests serve distinct purposes that go beyond the scope of black box and white box tests. Integration tests ensure the cooperation of different software components, while acceptance tests verify its suitability for real-world scenarios. Ensure that the software not only functions correctly but also effectively address users' needs. The combination of black box, white box, integration, and acceptance tests forms a comprehensive testing strategy that maximizes software reliability and user satisfaction.