# 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**: **SFT221 NEE group1**

**Members Present**:

|  |  |
| --- | --- |
| 1. Kim Ming Chau | 4. Gordon Tan |
| 2. Hak Kan Poon | 5. Yuhong Fan |
| 3. Elvin Karikari | 6. |

## 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** |
| **HK Poon** | **White box test case – TB01 & TB02 and partial scrum report Test execution on TB01 – TB03, TW01 – TW02. Debug on TB01** |  |
| **Gordon Tan** | **Implemented the change in function** |  |
| **Kim Ming Chau** | **White box test case – TW04, TW05 & TW06** |  |
| **Elvin Karikari** | **Implemented the change in function, scrum report** |  |
| **Yuhong Fan** | **White box test case – TW08, TW09, TW10** |  |
|  |  |  |
|  |  |  |

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** | **Some unit test cases haven’t been implemented** |
| **Reason for delay or block** | 1. **Some cases could not be compiled.** 2. **2. Due to misunderstanding, some cases have not been coded.** |
| **Impact on Project** | **Not much** |
| **Solution or work-around** | **Would implement to integration test and also test them in this milestone (5).** |
|  |  |
| **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 |
| Review MS4 | **Something that didn’t go well in MS4 and plan to improve it MS5.** | **Finished the left behind tasks in MS4 by 2 days and then start MS5.** |
| Program flow | **Gordon brief teams the revised structure.** | **Make use of the updated function for integration test in MS5.** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**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 |
| Finish the left behind tasks in MS4 | The correctness of function at Black box and white box testing is critical before going into integration test. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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? |
| **HK Poon** | **Integration Test TI01 & TI05** | **3 hr** | **Complete** |
| **Gordon Tan** | **Implemented the change in function & Integration Test TI03 & TI04 && TI08** | **5 hr** | **Complete** |
| **Kim Ming Chau** | **Integration Test TI02** | **2 hr** | **Complete** |
| **Elvin Karikari** | **Integration Test TI06, scrum report and reflection** | **3 hr** | **Complete** |
| **Yuhong Fan** | **Integration Test TI07** | **2 hr** | **Complete** |
|  |  |  |  |
|  |  |  |  |

**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 |
| HK Poon | Updated test matrix |
| Kim Ming Chau | Final test report |
| Gordon Tan | Test Execution |
| Yuhong Fan | Debugging |
| Kim Ming Chau | Git usage |
| HK Poon | Jira usage |
| Elvin Karikari | Scrum report & reflection |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Major Outcomes of Meeting:**

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

|  |  |
| --- | --- |
| Outcome | Impact on Project |
| **Tasks not completed during previous milestone** | **Allowed us to push forward for integration testing** |
| **Integration tests have completed** | **We are now able to move onto acceptance testing** |
|  |  |
|  |  |
|  |  |
|  |  |

**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 |
| **Quality of work** | **Everyone is now clear of project expectations and each member is more comfortable speaking up when unsure of what to do.** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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?

When encountering a test failure through automated testing and Git hooks within our group project, we collectively decided to adopt a structured process. We investigate the root cause of the failure, make essential code adjustments, conduct local retesting to prevent potential issues, and subsequently commit and push the revised code.

In projects involving continuous integration, the system automatically handles the building, testing, and integration of modifications. Changes that receive approval undergo rigorous testing and code review before being merged. Documentation is utilized to record the issue, the solution implemented, and the insights gained.

For intricate challenges, we're open to conducting post-mortem analyses. The primary objective of this methodology is to guarantee a dependable and high-quality codebase throughout the entirety of our development efforts.

1. Explain why we are automating the testing process and what the advantages of this automation are.

Automating the testing process offers several significant advantages for our project. Firstly, it enhances the efficiency of our development cycle by promptly identifying issues and failures, enabling timely corrective actions.

Secondly, automation ensures consistent and thorough testing across different environments, reducing the likelihood of discrepancies arising due to human error.

Additionally, automated testing helps maintain code integrity during rapid iterations and integrations, contributing to a more stable codebase. Moreover, it frees up valuable human resources from manual testing, allowing developers to focus on more creative and complex problem-solving tasks.

Ultimately, by automating our testing process, we aim to achieve higher software quality, faster development cycles, and an overall smoother collaborative workflow."

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?

In our development process, we found that writing integration and acceptance tests posed a slightly greater challenge compared to black box and white box tests. The complexity primarily stemmed from the need to ensure seamless interactions among different components and modules of the system, as well as verifying if the overall system met the desired user requirements.

Integrating various functionalities and simulating real-world scenarios for acceptance testing demanded a more holistic understanding of the application's behavior.

While the integration and acceptance tests required careful consideration of multiple interconnected components, we did end up writing more white box and black box tests. These types of tests allowed me to extensively examine individual units and functional aspects of the codebase. They were relatively quicker to implement and provided valuable insights into specific functionalities and code paths.

However, we recognized the critical importance of integration and acceptance tests for a comprehensive evaluation of the system's performance and its alignment with user expectations. Balancing the types of tests contributed to a well-rounded testing strategy that ensured both the micro and macro aspects of the application were thoroughly validated."

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 testing the interactions and interfaces between different components of a system. They check if individual pieces of code work correctly when combined together. Even if each component could pass both black box and white box testing, there may be unexpected issues when they interact with each other.

Acceptance tests focus on the overall behavior and functionality of the software from an end-user perspective. These tests validate that the system meets the specified requirements and that it fulfills its intended purpose. Acceptance tests cover user workflows and scenarios, ensuring that the entire system works as expected in real-world scenarios.

In short, while black box tests, and white box tests are crucial for verifying individual components and their behavior, integration and acceptance tests are essential for validating the system, ensuring its functional correctness, and detecting issues that can only be identified at higher levels of testing. These testing levels complement each other to create a comprehensive and robust testing approach.