# 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**: \_\_\_\_3\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Members Present**:

|  |  |
| --- | --- |
| 1.Chang Cui | 4. Xiaofei Xu |
| 2.Lok Yin Tai | 5. Xinyang Wu |
| 3.Siu Man Cheng | 6. Ye Tian |

## 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** |
| Chang Cui | Reflection  Jira Board Management  GitHub Repository Management  Update test Traceability Matrix | N/A |
| Lok Yin Tai | Blackbox tests implementation, execution, and debug  Update test Traceability Matrix  Jira Board Management  GitHub Repository Management | N/A |
| Siu Man Cheng | Whitebox tests implementation, execution, and debug  Jira Board Management  GitHub Repository Management | N/A |
| Xiaofei Xu | Whitebox tests  Update test Traceability Matrix  Jira Board Management  GitHub Repository Management | N/A |
| Xinyang Wu | Functions implementation  Jira Board Management  GitHub Repository Management | N/A |
| Ye Tian | Functions implementation  Jira Board Management  GitHub Repository Management | N/A |

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

**Summary of Meeting:**

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

|  |  |  |
| --- | --- | --- |
| Topic | Discussion Summary | Outcome |
| Review on last week work | The team reviewed the work and difficulties from the previous week. Team members reflects and comments to help each other. | Identified areas for improvement |
| Improvement base on professor comment and grading | Read the comments and discussed how to improve to do better for upcoming Milestones | Agreed to make more comments and communication in Jira to facilitate the communication. Use hook to do the automation tests. |
| Discussion on MS5 task distribution | The team discussed the tasks required for MS 5 and assign the tasks evenly. | The tasks are assigned to team members according to the member’s strength and knowledge evenly |
| Knowledge sharing on MS5 tasks | The team discussed the requirements and expectations for each deliverable. | All agreed with the delivery standards to continue the works |
| Discussion on deliverables dependency and due dates | The team discussed the dependencies between each task, deliverable standard and determined the due dates for each member. | All team members agreed with the due dates and ensured the high-quality completeness of 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 |
| separated the MS5 tasks into six categories depending on their burden and difficulty. | ensures that tasks are distributed evenly among team members, considering their specific skills and qualifications. |
| Distributing tasks via a random dice game | To eliminate bias in the assignment of tasks, introduced a fair and impartial method for assigning tasks. |
| Usage of the traceability | Emphasize that the traceability matrix shows the requirement number not the description of the requirement |
| More frequent meetings, more offline communication | Solve problems more efficiently and avoid communication barriers and misunderstandings. Reach consensus when there are disputes by communication and meeting together |
| Clarified the milestone deliverables and document to be submitted | Specified deliverables and documentation to be submitted in connection with milestones |
| Set up a due date for team documents review and feedback | As the load of this milestone is heavy, it is best to have a good time-management strategy. A buffer period for the evaluation and feedback of team documents, with a view to make early progress and improvement has been established. |

**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? |
| Ye Tian | Integration tests | 10 hours | Yes |
|  | Debugging | 10 hours | Yes |
|  | Jira Board Management | 30 mins | Yes |
|  | GitHub Repository Management | 30 mins | Yes |
| Lok Yin Tai | Acceptance test | 2 hours | Yes |
|  | Integration tests | 18 hours | Yes |
|  | Jira Board Management | 30 mins | Yes |
|  | GitHub Repository Management | 30 mins | Yes |
| Siu Man Cheng | Test Execution | 10 hours | Yes |
|  | Integration tests | 10 hours | Yes |
|  | Jira Board Management | 30 mins | Yes |
|  | GitHub Repository Management | 30 mins | Yes |
| Xiaofei Xu | Integration tests | 19 hours | Yes |
|  | Test Traceability Matrix | 1 hour | Yes |
|  | Jira Board Management | 30 mins | Yes |
|  | GitHub Repository Management | 30 mins | Yes |
| Xinyang Wu | Scrum Report and reflection | 3 hours | Yes |
|  | Hook setting | 2 hours | Yes |
|  | Test Execution | 15 hours | Yes |
|  | Jira Board Management | 30 mins | Yes |
|  | GitHub Repository Management | 30 mins | Yes |
| Chang Cui | Debugging | 10 hours | Yes |
|  | Integration tests | 10 hours | Yes |
|  | Jira Board Management | 30 mins | Yes |
|  | GitHub Repository Management | 30 mins | Yes |

**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 |
| Ye Tian | * Integration test case documentation * Debugging * Jira Board Management * GitHub Repository Management |
| Lok Yin Tai | * Acceptance test * Integration test code implementation * Jira Board Management * GitHub Repository Management |
| Siu Man Cheng | * Test Execution * Integration test case documentation * Jira Board Management * GitHub Repository Management |
| Xiaofei Xu | * Integration test * Update test Traceability Matrix * Jira Board Management * GitHub Repository Management |
| Xinyang Wu | * SCRUM report and reflection * Test execution and usage of hook * Jira Board Management * GitHub Repository Management |
| Chang Cui | * Debugging * Integration tests * Jira Board Management * GitHub Repository Management |

**Major Outcomes of Meeting:**

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

|  |  |
| --- | --- |
| Outcome | Impact on Project |
| Fully understanding of component, dependencies, and due date for each part | This understanding facilitates smoother collaboration and efficient task coordination throughout the Milestone 5. |
| Equitable Workload Distribution and job Rotation | The team made sure everyone had a fair share of the workload by agreeing on how to divide tasks and rotating them regularly. This approach made everyone feel like they owned their responsibilities and contributed equally. The positive team environments enhanced motivation and commitment to project success. |
| Integration tests implementation | Ensure that multiple functions work cohesively together to achieve the desired outcomes. This helped identify and resolve any potential integration issues. |
| Effective debugging | It helped improve the overall quality and reliability of the project. |
| Acceptance test preparation | It ensures the project met user requirements effectively and the final deliverables align with the expected outcomes. |
| Test Traceability Matrix Completion | It established a clear link between requirements, test cases, and test results. It validated the project's functionality against specific requirements. |

**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 |
| Active participation | Each team member participates in the scrum meeting actively. It allows effective communications, progress updates and task assignments |
| Time management | The team efficiently utilized the available time, clear the task due date, and arrange the working schedule accordingly |
| Collaboration and Cooperation | The positive team dynamics fostered collaboration and cooperation among members, promoting a supportive and productive working environment. |
| Active problem solving | Encouraged teamwork and collaboration in problem-solving to achieve positive team environment and improve work efficiency |

**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 |
| N/A | The meeting went smoothly. All tasks and questions were well discussed. Team members participated and discussed actively during the meeting. |

**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, we found some of the tests failed and prevented us from pushing our code to the repository. We found out the failed test case, corrected them, double checked if there is any bug and fixed it. We modified the code and made sure all the tests passed.

In the GIT hook, we automate testing to ensure that every code change goes through a series of tests before being pushed to the repository. If any of the tests fail, the hook prevents the code from being pushed, thus preventing potentially faulty or unstable code from being merged into the main branch. If a test fails, we handle the situation by reviewing the failed test results and identifying the root cause of the failure. We then make adjustments to the code, fix the issues, and re-run the tests. Once all tests pass, we can safely push the code to the repository.

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

We perform automated testing because it streamlines the development process, enhances code quality, and empowers teams to deliver reliable software products efficiently.

Automating the testing process has several key benefits:

(1) Faster Feedback: With automated tests, we get quick results on code changes, helping us detect issues early in the development cycle.

(2) Consistency: Automation ensures that the same tests are consistently applied to each code change, reducing the chance of human errors, and ensuring uniform testing across the codebase.

(3) Continuous Integration: Automated testing is vital for continuous integration, enabling us to seamlessly integrate code changes, identify problems early, and maintain a stable codebase.

(4) Increased Test Coverage: We can run a wide range of tests, including unit, integration, and acceptance tests, covering various scenarios, and improving overall test coverage.

(5) Time Efficiency: Automation saves QA time, as they no longer need to manually run tests for every code change. This also allows the developers to focus on critical development tasks.

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?

Integration and acceptance tests are more challenging to write than black box and white box tests for several reasons:

(1) Complexity: Integration tests contain several functions interacting with one another, making it more difficult to build up test environments and successfully handle dependencies.

(2) Real-World Scenarios: Acceptance tests are designed to simulate real-world user interactions, necessitating a thorough grasp of user needs as well as the system's behavior under various scenarios.

(3) Higher degree of Abstraction: Because integration and acceptance tests operate at a higher degree of abstraction, their complexity while developing and implementing test cases can grow.

(4) Greater Effort: While black box and white box tests can be created early in the development process, integration and acceptance tests require a more thorough evaluation of system functionality. This may take longer to create and implement.

We wrote more white box tests and black box tests compared to integration and acceptance tests, as unit tests target specific functions or components, providing rapid feedback during development. Integration and acceptance tests are more focused on validating system behavior as a whole, which requires careful planning and execution.

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.

Despite passing black box and white box tests, it is still necessary to write integration and acceptance tests for several reasons:

System-Level Verification: Integration tests ensure that different functions interact correctly with each other and that the integrated system functions as expected as a whole.

User Requirement Validation: Acceptance tests validate the system against user requirements, ensuring that it fulfills the intended functionality from an end-user perspective.

Complex Interactions: Some issues may only arise when components interact, making integration testing crucial to detecting such problems.

Real-World Scenarios: Acceptance tests simulate real user interactions, verifying that the system meets real-world use cases and scenarios that may not be covered by lower-level tests.