# 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. Jenna Moon | 4. |
| 2. Hyunjoo Han | 5. |
| 3. | 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 4 days after your lab day:**

* Integration tests document stored in repository with at least 4 sets of distance test cases (each case must have at least 4 distinct test data).
* Integration tests coded (store in repo), executed (results in Jira and in test documents) and debugged.
* Finish implementing/coding whitebox tests. Store in repo, executed, results in Jira (and on corresponding test documents, and debugged.
* Acceptance tests written and stored in repository.
* Updated requirements traceability matrix stored to the repository.
* Completed scrum report including reflection questions answered.

**Rubric:**

|  |  |  |
| --- | --- | --- |
| **Individual** | Group participation (includes GitHub commits and Jira usage) | 80% |
| Teamwork | 20% |
| **Group** | Integration test case document (well written, complete, good test data) | 10% |
| Integration test code (well designed and documented) | 10% |
| Finish coding all functions and main (well-designed, written, and documented) | 10% |
| Finish coding blackbox and whitebox cases (well-designed, written, and documented) | 10% |
| Acceptance tests (well-designed, written and documented) | 5% |
| Requirements traceability matrix updated | 5% |
| Test execution (performed, results recorded, issues created) | 5% |
| Debugging (bugs fixed, documented, Jira updated) | 5% |
| Git usage (used properly with good structure) | 5% |
| Jira usage (creates issues, tracks progress) | 10% |
| Scrum report & reflections | 20% |
| Meets deadlines | 5% |

**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** |
| **Hyunjoo Han** | **integration testing/unit testing automation/finish coding/acceptance testing/scrum report and reflection/Jira** | **None** |
| **Jenna Moon** | **integration testing/Requirement traceability matrix/scum report and reflection/Jira** | **None** |
|  |  | **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** | **Function implementation fix match took some time** |
| **Reason for delay or block** | **N/A** |
| **Impact on Project** | **N/A** |
| **Solution or work-around** | **Putting time and figuring it out** |
|  |  |
| **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 |
| Final Report | **Discuss the method of Final Report** | **Planned and assigned** |
| Acceptance test | **Discuss the method of Acceptance Test** | **Planned and assigned** |
| Test execution | **Discuss the method of Test execution** | **Planned and assigned** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**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 |
| Final testing report. | Ensure a comprehensive overview of the testing process, outcomes, and any identified issues. |
| Execute acceptance tests and debug. | Ensure that the software meets user requirements, identifying and resolving any issues that may impact the functionality or user experience. |
| Requirement Traceability Matrix | Ensure Requirement Traceability is up to date and reflected on the Final Testing Report |
|  |  |
|  |  |
|  |  |
|  |  |

**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? |
| Hyunjoo Han | Execute acceptance tests and debug, Final testing report, Jira and GitHub Management. | **2 hours** | **Yes** |
| Jenna Moon | Requirement Traceability Matrix and scrum report, Jira and GitHub Management. | **2 hours** | **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 |
|  | N/A- Milestones are done |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Major Outcomes of Meeting:**

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

|  |  |
| --- | --- |
| Outcome | Impact on Project |
| Confirmation on Acceptance testing | **Executing program** |
| Jira and GitHub Management | **Organize the project** |
| Final Report Discussed | **Final Report Planned and outlined. The purpose of the Final Report was discussed.** |
|  |  |
|  |  |
|  |  |
|  |  |

**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 |
| Acceptance testing | **Executed successfully to fulfill the purpose of the project** |
| Requirement Traceability Matrix | **Requirements are up to date and tests are fulfilled to meet the requirements.** |
| GitHub | **Useful for version control and keeping track of changes** |
| Jira | **All contributed and tasks are organized** |
|  |  |
|  |  |
|  |  |

**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 |
| Absence of members | **Busy schedule and communication problems** |
| Limited time | **Other work/assignment commitments with limited time** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Reflections**:

Answer the following questions using your own words. Make sure that each answer comprises a minimum of 100 words.

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?

We use Git hooks for testing automatically in our workflow, which helps keep our code reliable. Sometimes, tests find issues, making it tricky to push code to the repository smoothly. To deal with this, we quickly tackle the problems using a systematic approach. Git hooks notify us about test results, and we check them using GitHub Desktop.

When a test fails, our team works together to figure out why. We thoroughly look into the problem and fix it, making sure our code stays strong and meets our testing standards. After solving the issues, we continue with the Git push. We also make a point to write clear messages about the changes we made. This helps us keep track of what we did, whether it's fixing tests or making big improvements to the code. While we like the efficiency of automated hooks, we also make sure to keep a balance that lets developers work flexibly. This shows our commitment to keeping control of our code versions and tracking progress in a way that's both high-quality and efficient.

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

Automating testing in software development offers numerous advantages that enhance the efficiency and reliability of the development process.

Firstly, automated testing allows developers to check their code frequently and rapidly. This is particularly beneficial during continuous integration and delivery, ensuring that changes to the codebase are promptly validated, reducing the chances of introducing errors. Secondly, automated testing is an effective means of catching bugs and defects early in the development of a lifecycle. Continuous and automated testing helps identify issues swiftly after code changes, reducing the likelihood of bugs lingering until later stages of development. Third, automated testing acts like a safety net for deploying new features, giving developers confidence in rolling out changes. Having a thorough set of automated tests allows developers to make changes and modify the code confidently, ensuring that everything still works well. Additionally, automation also allows teams to focus more on improving software quality. By automating repetitive testing tasks, developers can allocate more time to creative and exploratory testing, ultimately enhancing overall software quality.

Automated tests ensure consistency in the testing process by performing tests in the same way every time. This eliminates human errors associated with manual testing, creating a reliable and reproducible testing environment. In summary, automated testing streamlines the testing process, contributing to higher efficiency, reliability, and confidence in the software development lifecycle. The result is higher-quality software delivered more quickly.

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?

Writing integration and acceptance tests can often be more challenging than black box and white box tests for several reasons. Integration tests involve checking how different components work together as a whole, which requires a comprehensive understanding of the entire system. This can be challenging because it necessitates knowledge of the relationships between different modules and how they interact.

Acceptance tests, on the other hand, focus on ensuring that the software meets the specified requirements and satisfies user expectations. This requires not only technical knowledge but also a deep understanding of user needs and behaviors. It's often harder to predict all the ways users might interact with the system, making it challenging to cover all scenarios in tests.

Moreover, writing integration and acceptance tests can take longer due to the need for setting up realistic scenarios and dealing with the complexities of the entire system. In contrast, black box tests focus on inputs and outputs, and white box tests delve into the internal logic of individual components, making them more modular and easier to write.

In summary, the challenges in integration and acceptance testing stem from the need for a holistic understanding of the system, its interactions, and meeting user expectations. Balancing the depth and breadth of test coverage in these tests requires a careful and thorough approach.

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.

Even though black box and white box tests are important for checking individual parts of software, it's still necessary to do integration and acceptance tests because they bring special benefits that go along with those earlier tests. Integration tests make sure that different parts, which passed the earlier tests, can work together without any problems. These tests find hidden issues, mistakes, or connections that show up when different parts of the software talk to each other, making sure everything runs smoothly in real-life situations.

Now, acceptance tests, like Alpha and Beta testing, involve real people who use the software. These tests help find out more than just technical problems. They show if the software is easy for people to use, if there are any issues with how it looks, or if it's not working fast enough. Including real users in these tests makes sure the software not only does what it's supposed to technically but also meets what users expect. This makes the software better overall and more ready for the market.

In short, integration and acceptance tests add on to what we've already checked with black box and white box tests. Integration tests focus on making sure different parts work well together, and acceptance tests check if users are happy and if the software is easy to use, reducing problems and making the software more reliable.