# **MILESTONE 3** -- 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**: \_\_\_4\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Members Present**:

|  |  |
| --- | --- |
| 1. Anna Francesca Dela Cruz | 4. Gulpreet Kaur |
| 2. Irish Banga | 5. |
| 3. In Tae Chung | 6. |

## Milestone 3 Tasks

In this milestone you will create issues to design the functions, design all of the functions you need to complete the project and store the specifications in the repository. As soon as the specifications start to be produced, you can start to design the blackbox tests (what they test, how to perform them and test data). Once tests are written, they can be implemented and added to the repository and any team members not otherwise busy can start to implement the functions. You will also build a function-test matrix that shows the blackbox tests for each function. This will be maintained through the testing cycle as new tests are added.

**Deliverables Due at end of Lab:**

* Completed SCRUM report and reflections

**Deliverables Due at 23:59 6 Days after Lab:**

* A set of function specifications stored in the repository,
* A set of blackbox tests as test documents with test data for the functions.
* Start writing blackbox test code and store in repository. (at least 1 required)
* Start implementing functions and store in repository. (optional)
* A function-test matrix added to the repository.
* Updated Jira project to show activities and progress.

**Rubric**

|  |  |  |
| --- | --- | --- |
| Individual | Group Participation | 75% |
| Teamwork | 10% |
| SCRUM Report | 15% |
| Group | Function Specs (documented, correct, complete, well-written) | 20% |
| Test documents (well-written, complete, good test data) | 20% |
| Test Code (well-designed, written and documented) | 10% |
| Git Usage (used properly with good structure) | 5% |
| Jira Usage (creates issues, tracks progress) | 10% |
| Meets Deadlines | 10% |
| SCRUM report & reflections | 25% |

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

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 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

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

**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? |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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

**Major Outcomes of Meeting:**

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

|  |  |
| --- | --- |
| Outcome | Impact on Project |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

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

**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. In this milestone, we write the blackbox tests but not the whitebox tests. Explain why we can write the blackbox tests but not the whitebox tests.   
     
     
   In this case, where you lack access to the necessary code or internal information, it is indeed advisable to concentrate on building blackbox tests rather than whitebox tests. A justification for why blackbox tests can be created in this scenario but whitebox tests cannot is provided below:
2. Blackbox testing: Blackbox testing isolates the system's exterior behaviour and does not demand understanding of its internal implementation. The system is viewed as a "black box," where the tester can only interact with the inputs and watch the replies or outputs. Blackbox testing can be carried out without having access to the code or internal details because it only looks at outwardly visible behavior. Based on the system's anticipated functionality, usability, performance, or compatibility with various inputs or scenarios, you can develop test cases.
3. Whitebox Testing: Whitebox testing, on the other hand, necessitates familiarity with the system's underlying operations, including its code and organizational structure. It entails looking at the code, comprehending control processes, and spotting any problems or weaknesses. Whitebox testing is ineffective without access to the appropriate code or internal information. Whitebox testing methods that rely largely on internal structure knowledge to provide extensive test coverage include code coverage analysis and structural testing.  
   In conclusion, it is fair to concentrate on blackbox testing given that you lack access to the required code or internal data. Without having knowledge of the technical implementation, blackbox tests can be created and run based on the system's externally observable behaviour. With this method, you can test the system's performance, usability, compatibility, or usefulness without having to deal with the technical details.
4. Explain why we need the function-test matrix and why it is important in a large project.

The function-test matrix is necessary, especially in large-scale projects, for the following reasons:

* 1. Ensures every required aspect of the program is complete and tested: The matrix tracks each requirement and ensures that the corresponding test cases have validated them. It helps developers and testers see gaps during the testing process.
  2. Makes managing the number of test cases more efficient: Due to the number of test cases a large project may have, managing them can be difficult. The matrix allows you to organize and track test cases by providing a systematic approach.
  3. Streamlines collaboration and communication: Because everyone in the team has access to the matrix, it allows team members to see the progress of the project and its requirements at all times. It promotes a shared understanding among the team about the tests that have been done and need to be done.
  4. Helps prevent potential risks: Because the matrix provides a shared understanding of the project’s progress, the team can better determine what tasks and resources to prioritize in order to satisfy all project requirements. This can prevent delays and failures in meeting project deliverables.

1. Other life cycle models left team members idle while waiting for parts of the project to be completed. Describe how an agile model, like the one we are using, avoids this problem and keeps the whole team busy all the time. Does this make managing the project simpler or more complex and why?

To prevent team members from being inactive, agile models like Scrum Report and reflection questions encourage iterative and incremental progress. Here is how an agile paradigm affects project management and helps to keep the whole team busy:

1. The test plan is a document that explains how we will test a product. It includes things like our strategy, goals, scope, and what resources we'll need. In agile projects, the test plan can change over time as the project progresses. It helps the team know how to do the tests well and manage them effectively. It also makes sure that testing is done in a way that matches the way we're developing the product.
2. The Scrum report gives information about how the project is going. It tells us about the tasks that have been done and what still needs to be done. It also tells us about any problems or obstacles that we're facing. The report helps the team and stakeholders understand the project's status easily. By looking at the report regularly, the team can find any issues or delays and solve them quickly. They can use the information in the report to make decisions based on data, which helps keep the project on the right path.
3. Test cases are important to make sure the final product is of good quality. They describe the steps, expected results, and requirements for specific tests. The team can use these test cases to guide their development process while writing code and make sure that the features they create work as expected. In agile development, where the focus is on delivering functional parts with built-in quality assurance, writing code and test cases are done together.
4. By discussing these reflection questions, the team can gain insights into their performance, methods, and how they work together. The main goal is to focus on recognizing achievements, identifying areas for improvement, and finding practical solutions to any issues that arise. This ongoing process of reflection and adjustment has a direct impact on the team's progress, efficiency, and overall performance in future sprints or iterations.