# **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**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Members Present**:

| 1. Jubril Olawale Akolade 167529213 | 4.Iraklis Tsanachtsidis 122226228 |
| --- | --- |
| 2. Frank Prerez 141647222 | 5.Aum Rasikbhai Parsana 112872221 |
| 3. Tarun Thomas 113605224 | 6.Rutarj Mrushad Shah 170870216 |
| 7. Faaz Sherwani 113026223 |  |

## 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** |
| --- | --- | --- |
| Jubril Olawale Akolade | **Scrum Report** |  |
| Iraklis Tsanachtsidis | **Scrum Report** |  |
| Tarun Thomas | **Black box testing** |  |
| Faaz Sherwani | **Function definition and black box test** |  |
| Aum Rasikbhai Parsana | **black box testing** |  |
| Rutarj Mrushad Shah | **Black box test** |  |
| Frank Prerez | **Jira assignment** |  |

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** | **None** |
| --- | --- |
| **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 |
| --- | --- | --- |
| Milestone 3 | **Discussion and planning** | **Success and completion** |
| Team meeting | **Reviewing completed tasks and identifying issues or delays** | **Success** |
| Functions | **Established the process for documenting function specifications** | **Success** |
| Black box | **writing of black box texts** | **Success** |
| Plans for milestone 4 | **Discussed and planned approaches for the next milestone** | **Success** |
| Jira and scrum | **Both were planned and completed** | **Success** |
|  |  |  |

**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 |
| --- | --- |
| Finalized the architectural design for the project. | Finalizing the architectural design is crucial to ensure a solid foundation for the project. It helps determine the overall structure, components, and technologies to be used. By finalizing the architectural design, the team can align their efforts, make informed decisions, and proceed with the development process smoothly. |
| Prioritized tasks based on dependencies and criticality | Prioritizing tasks is essential to maximize efficiency and meet project goals effectively. By considering dependencies and criticality, the team can identify tasks that need to be completed first to avoid delays or roadblocks. This prioritization ensures that the most critical and interdependent tasks are addressed promptly, allowing for a smoother progression of the project. |

**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? |
| --- | --- | --- | --- |
| Jubril Olawale Akolade | **SCRUM Report** | **1 hour** | **Yes** |
| Iraklis Tsanachtsidis | **SCRUM Report** | **1 hour** | **Yes** |
| Tarun Thomas | **Writing Black box tests** | **2 hours** | **Yes** |
| Faaz Sherwani | **Function implementation** | **4 hours** | **Yes** |
| Aum Rasikbhai Parsana | **Writing Black box tests** | **2.5 hours** | **Yes** |
| Rutarj Mrushad Shah | **Writing Black box tests** | **2 hours** | **Yes** |
| Frank Prerez | **Jira organization and assignment** | **1 hour** | **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 |
| --- | --- |
| Rutarj Mrushad Shah, Tarun Thomas  &Rutarj Mrushad Shah | Implement Functions |
| Faaz Sherwani &  Tarun Thomas | Write blackbox tests for Function B |
| Jubril Olawale Akolade  &Iraklis Tsanachtsidis | Refactor existing code for better modularity |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Major Outcomes of Meeting:**

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

| Outcome | Impact on Project |
| --- | --- |
| Completed SCRUM report | **Completion of SCRUM Report and documentation of the teams’ moves** |
| Established clear goals and assigned tasks for the next week to each member | **Planning and organization that allows the team to work efficiently** |
| Made progress in function declaration, documentation and blackbox test writing. | **Completed the tasks of function declaration and writing of black box tests** |
|  |  |
|  |  |
|  |  |
|  |  |

**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 |
| --- | --- |
| Functions | Effective communication and collaboration among team members. |
| Work division | Clear understanding of individual responsibilities and progress made. |
| Scrum Report | Timely completion of the SCRUM report and reflections. |
|  |  |
|  |  |
|  |  |
|  |  |

**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 |
| --- | --- |
| Task assignment | The discussion on task priorities and allocation of workload became challenging, requiring additional time and effort to reach consensus among team members. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Reflections**:

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 milestone, we focus on writing blackbox tests rather than whitebox tests due to the stage of the project and the current goals.

Blackbox testing is a type of testing that focuses on the external behavior and functionality of the system under test without considering its internal implementation details. It treats the system as a "black box" where only the inputs and outputs are observed and verified. Blackbox tests are written based on the system's specifications and requirements, allowing us to validate if the system meets the desired functionality from an end-user perspective.

On the other hand, whitebox testing involves testing the internal structure, code, and logic of the system. It requires knowledge of the system's implementation details and typically involves techniques such as code coverage analysis, unit testing, and structural testing. Whitebox tests

are closely tied to the specific implementation of the system and are typically written by developers to ensure code correctness, identify bugs, and cover edge cases.

In the milestone described, the emphasis is on designing the functions, creating function specifications, and documenting them in the repository. Since the focus is on defining the desired behavior and functionality of the functions, it aligns well with writing blackbox tests. Blackbox tests can be written based on the function specifications, test data, and expected outcomes without delving into the internal implementation details.

Writing whitebox tests would require a deeper understanding of the code implementation, which might not be the immediate priority at this stage. It is more efficient to focus on completing the function specifications, designing the blackbox tests, and implementing them to verify the desired behavior. Once the system's functionality is established and the code implementation is more stable, whitebox testing can be pursued in subsequent milestones to ensure code coverage, handle edge cases, and perform more detailed checks on the internal workings of the system.

2. Explain why we need the function-test matrix and why it is important in a large project.  
  
A function-test matrix is a tool used to map and track the relationship between functions or features of a system and the corresponding blackbox tests. It provides a systematic overview of which tests cover which functions and helps ensure comprehensive test coverage throughout a large project. Here are some reasons why a function-test matrix is important in a large project:

* Test Coverage: In a large project, there can be numerous functions or features that need to be tested. The function-test matrix helps ensure that all functions are adequately tested by mapping each function to the relevant blackbox tests. It acts as a visual representation of the test coverage, allowing the project team to identify any gaps in testing and ensure that all functions are thoroughly validated.
* Traceability: The function-test matrix establishes a traceability link between the requirements, specifications, and tests. It helps ensure that each requirement or function has corresponding blackbox tests that verify its proper implementation. This traceability is crucial for compliance purposes, as it allows stakeholders to easily track the testing progress and verify that all requirements have been met.
* Test Management: In a large project with multiple team members, test management becomes crucial. The function-test matrix serves as a central repository for test information, making it easier to manage and organize the tests. It allows the team to track the progress of each test, identify any redundant or duplicate tests, and prioritize testing efforts based on critical functions or features.
* Collaboration and Communication: The function-test matrix serves as a communication tool among team members, testers, developers, and other stakeholders involved in the project. It provides a shared understanding of the testing scope, objectives, and progress. By having a clear overview of which tests cover each function, team members can collaborate effectively, discuss test results, and troubleshoot issues more efficiently.
* Regression Testing: In a large project with frequent updates and changes, regression testing becomes essential to ensure that existing functionality is not affected by new changes. The function-test matrix helps identify which tests need to be rerun or updated when changes are made to specific functions. It simplifies the process of maintaining and updating the test suite, reducing the risk of regression issues.

Overall, a function-test matrix plays a vital role in ensuring comprehensive test coverage, maintaining traceability, facilitating test management, enabling collaboration, and supporting efficient regression testing. It is an important tool in large projects to keep track of the testing efforts and ensure the quality and reliability of the developed system.

3. 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?

An agile model, such as the one being used, aims to keep the whole team engaged and productive throughout the project duration. It avoids the problem of team members being idle while waiting for parts of the project to be completed through several key practices:

* Iterative Development: Agile models adopt an iterative approach, where the project is divided into small increments called iterations or sprints. Each iteration typically lasts a few weeks and focuses on delivering a working product increment. This allows team members to continuously work on and contribute to the project, even if certain parts are still being developed or integrated. It avoids long periods of inactivity by providing a constant stream of work and progress.
* Cross-Functional Teams: Agile teams are typically cross-functional, meaning they consist of individuals with diverse skills and expertise. Instead of having specialized teams for different project phases, such as development, testing, or design, agile teams include members capable of working on multiple aspects of the project. This enables team members to collaborate and contribute in various areas, ensuring that there is always work to be done.
* Adaptive Planning: Agile models emphasize adaptive planning, where project plans and priorities are continuously reviewed and adjusted based on feedback and changing requirements. This flexibility allows the team to adapt to evolving project needs and make efficient use of available resources. If certain parts of the project are delayed or blocked, the team can reorganize and focus on other tasks that are ready for implementation, keeping everyone productive.
* Task Prioritization and Backlog: Agile teams maintain a prioritized backlog of tasks or user stories. This backlog contains a list of features, functions, or requirements that need to be implemented. Team members can always pull tasks from the top of the backlog, ensuring that they have work to do at all times. If one part of the project is delayed, team members can select tasks from other areas to work on, preventing idle time.

Managing the project in an agile model can have both simplified and complex aspects, depending on the context and team dynamics. Here are some factors to consider:

Simplified Management:

- Clear Visibility: Agile models provide transparent and real-time visibility into project progress, priorities, and impediments through tools like Kanban boards or task tracking systems. This transparency simplifies project management by allowing the team to identify bottlenecks, track individual contributions, and make informed decisions.

- Flexibility: Agile models offer the flexibility to adapt and adjust plans based on emerging requirements or changes. This adaptability simplifies management as the team can respond quickly to shifts in priorities or resource availability, ensuring that work is continuously progressing.

- Collaboration: Agile emphasizes close collaboration and communication among team members, stakeholders, and customers. Regular meetings, such as daily stand-ups or sprint reviews, promote effective coordination and decision-making, making it easier to manage the project as a whole.

Complex Management:

- Balancing Workload: Ensuring a balanced workload across team members and iterations can be challenging in agile models. The project manager or Scrum Master needs to carefully monitor the allocation of tasks, avoiding overloading or underutilizing team members.

- Changing Priorities: Agile models embrace changing priorities and requirements, which can introduce complexity in managing project scope and expectations. Continuous reprioritization and reevaluation of tasks may require ongoing communication with stakeholders to manage their expectations.

- Dependencies and Integration: Agile teams may face challenges when integrating components or functionalities developed by different team members simultaneously. Managing dependencies and ensuring smooth integration can be complex, requiring effective coordination and collaboration among team members.

Overall, while an agile model helps keep the team engaged and avoids idle time, managing the project can present both simplified and complex aspects. The iterative nature, adaptability, and emphasis on collaboration simplify certain aspects of project management. However, balancing workloads, addressing changing priorities, and managing dependencies can introduce complexity that requires careful attention and coordination.