# 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. Song Hwan Oh | 4.Sangjune Lee |
| 2. Shine Lee | 5. Ji Ho Nam |
| 3. Yoojin Lee | 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** |
| **ALL** | **Acceptance Testing , Integration Testing, Test Excution, SCRUM and Reflection** | **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 |
| Integration Testing | **Integration testing written and discussed** | **Completed** |
| Acceptance Testing | **Acceptance testing written and discussed** | **Completed** |
| Test execution | **discussed then executed** | **Executed** |
| SCRUM | **SCRUM Done** | **Completed** |
| Reflection | **Discussing Reflection question** | **Completed** |
| Git | **Git update to each branch (Debug tickets on Git project)** | **Completed** |
| Jira | **Task Schedule setup in Jira (Debug ticket)** | **Completed** |

**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 |
| Prioritization of tasks | Equal amount of works assigned to each member of team. |
| Acceptance Testing | Choosing Alpha and beta testing as process. |
| Integration Testing | Decision made on what are the “All components” for integration testing. Choosing Bottom – up testing as process. |
| Testing execution | Testing code has been executed and recorded |
|  |  |
|  |  |
|  |  |

**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? |
| ALL | **Scrum report** | **30min** | **Yes** |
| ALL | **Jira and Github Project page updated and assigned** | **30min** | **Yes** |
| ALL | **Discussion for next week tasks** | **15min** | **Yes** |
| ALL | **Test execution and discussion about debugging and assigning new tickets to members** | **1hr** | **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 |
| Sangjune Lee | Jira control |
| ALL | Meeting on Aug 7th Monday 9pm to 11pm |
| ALL | SCRUM , Reflection |
| ALL | Final test report |
| ALL | Test Execution() |
| ALL | Debuggings |
| ALL | Test Matrix |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Major Outcomes of Meeting:**

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

|  |  |
| --- | --- |
| Outcome | Impact on Project |
| Detailed improvement on functionality of program | **With multiple Black box and white box testing and Acceptance and integration testing, we have achieved a certain level of perfection on our new implemented Functions with data structure within the existing program.** |
| Confirmation on Acceptance testing | **Executing program individually (Alpha and beta) and listed out Debugs for matrix and ticketed to members** |
| Confirmation on Integration Testing | **Executing Testing codes together and listed out Debugs for matrix and ticketed to members** |
|  |  |
|  |  |
|  |  |
|  |  |

**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 |
| SCRUM | All contributed. |
| Git | **Useful for version control and keeping track of changes** |
| Meeting | All attended meeting. |
| Integration testing | **Made all tests passed** |
| Acceptance testing | **Choice or user and choice of alpha user done and documented** |
|  |  |
|  |  |

**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 | **N/A** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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?  
     
   We've taken steps to enhance our development workflow by introducing informative prompt messages. These prompts guide developers through the commit process in the Bash terminal, ensuring careful consideration before finalizing changes.

Currently, we utilize GitHub Desktop to monitor test outcomes, identifying any failures reflected in testing files. We promptly address these issues, maintaining a robust and reliable code base.

After resolving issues and implementing fixes, we proceed with the commit. We also acknowledge the significance of documenting progress, using commits to log significant advancements in files or the overall project, promoting transparency and collaboration.

While automated hooks offer efficiency, we're mindful of potential limitations on version control and progress tracking. Striking a balance between automation and developer flexibility remains a key focus.

In essence, our prompt messages and strategic use of GitHub Desktop underscore our commitment to code quality, thorough version control, and comprehensive progress tracking in our development process.

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

Automating the testing process brings multiple advantages to software development:

Efficiency: Automated tests swiftly detect defects and regressions, accelerating the feedback loop.

**Consistency**: Precise test script execution eliminates human errors, ensuring reliable results.

**Coverage**: Automated tests comprehensively cover scenarios, including edge cases and configurations.

**Frequency**: Integration into agile and CI/CD pipelines allows frequent testing, catching issues early.

**Scalability**: Handling growing test cases efficiently suits evolving codebases.

**Collaboration**: Objective tests foster effective communication among stakeholders.

**Cost** **Savings**: Despite initial setup, long-term reduction in manual efforts yields cost benefits.

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 testing posed a greater challenge than white box testing due to the need for intricate connections between units and the creation of new testing code. Acceptance testing, particularly Alpha and Beta phases, brought further complexity, requiring careful user selection and feedback coordination.

In our team, identifying suitable Beta users from within our group proved intricate, considering real-world usage replication and constructive input.

Facing complexities, we intensified our focus on integration testing codes. Dissatisfied with initial black and white box testing outcomes, we collectively invested more effort into refining our testing processes, particularly in MS05.

This dedication aimed to comprehensively understand our software's behavior, ensure compatibility, and proactively address potential issues, aligning with our commitment to a robust end product. Our testing journey demonstrated our resolve to enhance strategies and deliver top-quality software.

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.

Black box and white box tests are vital for checking individual software parts, but integration and acceptance tests offer unique advantages that complement these stages.

Integration tests ensure different components, which passed earlier tests, work together without issues. They unveil hidden glitches, mismatches, or dependencies that emerge when parts interact. This guarantees seamless operation in real-world scenarios.

Acceptance tests (like Alpha and Beta testing) involve real users, revealing insights beyond technical correctness. They identify usability issues, user interface glitches, or performance bottlenecks that earlier tests might miss. Including real users ensures the software meets their needs, enhancing quality and market readiness.

In summary, integration and acceptance tests extend beyond black box and white box validations. Integration tests ensure smooth component interaction, while acceptance tests confirm user satisfaction and usability, minimizing defects and boosting software reliability.