CS 250

Deron Dantzler  
Charles Morris, Jr.

25 February 2024

Final Project: Sprint Review and Retrospective

When working on the SNHU Travel project and acting in the role of the scrum master, I had to establish the main goal for the initial sprint in collaboration with the other members of the team. It was important to conduct daily scrum meetings where I would ask team members the three key questions: “What did you do yesterday? What will you do today? Are there any impediments?” These daily meetings helped us track our overall progress and identify problems before they became serious. As tasks were completed, I updated the product backlog to ensure that everything required would be completed by the end of the sprint. After the sprint was over, I conducted a sprint review and retrospective that determined how much of our goal was achieved and how we could perform better next time. I also considered client feedback after the client was shown the work.

As the product owner, I gathered information on what the users and stakeholders wanted. Then, I created user personas and stories to help guide me when creating the product backlog and managing the project. User stories helped me create clear examples of what the users wanted, helped me prioritize the product backlog, and gave me acceptance criteria that were useful for creating testing and validation. It was also helpful to conduct regular interviews and meetings with the users to accurately assess their needs and clear up any ambiguities.

When developing tests as a tester, it was essential to carefully read the user stories, particularly the description of what the end user wanted. By having a complete and accurate description, I knew what to test to ensure that the developers implemented each feature correctly. If I did not have enough detail about how a user wanted a feature implemented, I would contact them for more information.

As a developer, I needed concrete examples of what features the users wanted and what the completed features should look like, so I asked the product owner to provide me with user stories. I also inquired about what features should be prioritized. I corresponded with the tester and asked them to create thorough tests to ensure that the features I implemented worked properly. To help the tester create their tests, I gave them documentation that described how the features worked, including code and the API.

The Scrum-Agile approach to the software development lifecycle helped me write user stories in a way that was specific to solving the user’s needs. User story value statements were created using the structure, “As a user, I want to perform some task so that I can achieve some goal.” This allowed me to clearly define each feature the user wanted implemented. Once the goal was defined, I estimated the size of each story and prioritized them. Then, I created acceptance criteria to help the developer and tester understand what exactly needs to be met for the feature to be considered complete.

When the client decided to update their requirements for the software, the Scrum-Agile approach allowed the team to make the required changes effectively without having the restart the entire project from scratch. For example, when the layout of the Java app was changed from a list view to a slideshow view, the agile approach allowed the scrum team members to coordinate when implementing the new feature. The product owner and scrum master updated the product backlog and set the new requirements, the developer revised the code, and the tester received new information to test that the app worked properly in slideshow format.

Communication among the scrum team members was essential for ensuring that everyone was on the same page. For example, when the developer was updating the app to focus on detox and wellness destinations, they had to email the product owner asking for more details through user stories along with what features took priority. The developer also had to ask the tester to create new tests to check that the updated features worked properly. If this communication had not occurred, the developer may have worked on implementing features that did not take priority, or the tester may not have been able to effectively test the new features, causing potential errors and development delays.

The scrum events ensured that the app was developed effectively. Sprint planning allowed the scrum master to plot out what features could be implemented within the given timeframe. Daily standups helped members relate their progress and any issues they encountered. The sprint review and retrospective allowed us to present our work to the stakeholders and receive feedback that we could use for our next sprint. It also allowed us to reflect on our sprint and find ways to improve our next sprint. Using tools like JIRA, a Kanban board software, helped allow the scrum team to coordinate their work. Team members could see when tasks were being worked on or completed in real-time. They could also add notes or comments on tasks to which other members could reply.

The agile approach was essential for developing SNHU Travel effectively with the events that happened. If the waterfall approach had been used, and the client decided that they wanted to change the requirements of the software mid-development, the software may have required a major rework, delaying development for months. Fortunately, agile allowed us to pivot quickly and implement the updates without starting over. The only downside to using agile in this context was that the team was used to the waterfall approach. Acclimating to the agile approach required time and learning. However, this commitment allowed the project to be developed less stressfully and in a shorter time than if the waterfall approach had been used. It also helped prove to ChadaTech that the agile approach is more effective than the waterfall approach, convincing them to shift all their development teams to the agile methodology.