Steven Cruickshank

February 19, 2022

Sprint review and retrospective:

This has been one of my favorite courses I've taken throughout my Computer Science degree thus far. It's so easy to get bogged down in coding, and theory with this degree. From the day-to-day to the big picture goals; For the first time I finally have a solid idea of how a development related job would operate. This curriculum seemed to come very easily to me given my experience working in a large-scale accounting office. Things like working with accounting software as an end-user, and regression testing are things I perform regularly during my day job. I feel this experience gave me a good business perspective when working with the case-studies in this course, which allowed me to excel.

One of my main take-aways from this course has been the role of the scrum as the backbone of development. Communication is important regardless of one's job, but scrum meetings are an incredible vehicle for the necessary communications required for a well-oiled development team. During my work for the SNHU travel project, it was interesting investigating various implementations of real world scrum meetings. I'm a little old fashioned, so I think I prefer a nice Kanban board, or even a whiteboard with in-person meetings every day.

Understanding the role of the Scrum Master for our project revealed just how much they are responsible for. In addition to facilitating communications via scrum meets, they're also making sure the development team's needs are fully met, whether that be new office equipment, or training in the latest programming language rollout.

Working as Product Owner was my favorite modules during this course. Our goal was to meet with end-users and communicate their needs to our development team, and stakeholders via user stories. With so much coding under my belt, I found the end-user's requests to be extremely relatable and easy to digest/communicate. After all, if your software product isn't easy to use, why should people use it? As product owner, it's important to ask follow-up questions of these end-users in order to get as many details as possible. If you're not communicating full details in these User Stories, you're just creating an endless loop of back-and-forth between your testers and the end-users to ask for more details.

As I mentioned, I'm very familiar with regression testing. My job requires twice-yearly regression tests in accordance with accounting software roll-outs. When taking the role of Tester, I felt very at home. While reading user stories for SNHU Travel, it was not difficult for me to think about follow-up questions, or understand the importance of each story's implementation. Given my experience with regression testing, I knew coming into this how much of a slog they can be to perform. During our group discussion for SNHU Travel, I assumed the role of Tester, and one of my first goals was to switch SNHU to all automated testing to free up enormous amounts of time for our team.

The SLDC was integral to completing SNHU Travel's user-story assignment. I would define both scrum meetings, and the Product Owner receiving/clarifying information with end-users as **planning** and **defining**. When filling out user-story requests, the Product Owner was fulfilling the **designing** and **building** requirements. Once those are finished, the Testers along with the Development Team enter the **testing** and **deployment** of those user-stories.

About half-way through the SNHU Travel project, the Product Owner informed us that there would be a major change in our software. Instead of being a generalized travel and booking

service, they wanted to focus strictly on health and wellness vacation packages. Had SNHU Travel been developed using the Waterfall method, this could have derailed everything we had worked on thus far. Given SNHU Travel adopted an Agile-Scrum SDLC, we were able to diffuse the situation, and delegate these changes to small work groups. This allowed us to only make small changes to the overall project, yet still accomplish this project shift.

I touched on it before, but I felt like my implementation requirements when assuming the role of Tester during our group work was straight-forward and easily understood. For instance I explained why I wanted to shift from manual regression testing to automated testing, and how that would be a benefit to our work team. I received some great feedback that built up on some of my original ideas which would have made them even more effective.

One of those most important Scrum-Agile principles employed was delivering an effective software product above all else. Any changes that may have happened weren't met with panic, but with open arms, which is another integral Scrum-Agile principle. As a scrum team we collaboratively built a product backlog, used examples of Kanban boards, criticized **ideas** and not people by suggesting ways we could build upon each other's planned implementations during group work. We were allowed to work in each Scrum-Agile role to contribute to the completion of the SNHU project, which gave the impression of contributing collaboratively even though a good portion was not actual group work.

Personally I loved the Scrum-Agile framework while working with the SNHU Travel project. It's easy to read texts and take notes, but I felt like I was thrust into each role in a functioning software project. Agile allowed us to constantly be communicating between Product Owners, end-users, and Testers. It allowed us to handle significant changes to the travel software more than halfway through the project. Agile made for a lot more collaborative work between

scrum teams, and even our group work assignment. I think the SNHU Travel project team was very effective, so I cannot think of any cons I experienced while working with Agile on this project. Theoretically, if your Scrum team lacked communication, or otherwise felt a sense of unfulfillment, I could see how things might derail. However I feel that could be the case regardless of Agile implementation. The project itself was only 8 weeks long, but we did encounter a few issues which lent itself well to SNHU Travel's adoption of Agile. Had we implemented the waterfall method, that big shift mid-development may have been a **much** bigger issue.