Sprint Review and Retrospective

Throughout the development of the SNHU Travel project, every role contributed to the successful outcome. In the early stages of development, the project owner communicated with the client about the requirements for the product. They also communicated with the consumers on what they expect from the product and how they would interact with it. This requirement collecting phase was vital to later stages of development because it established the understanding of the requirements which eventually became tasks that the team prioritized and placed on the scrum board. The requirements gathering phase sets the foundation for the rest of planning and allows the team to make a more accurate estimate of how much they can get done in the given time. A little while into development, the product owner and the client discussed a change in requirements; the client decided the product needed to follow the current travel trend of detox and wellness. The product owner communicated with the team about the new requirements and how they should proceed with the given timeline. Since this is agile development, it was expected that the original timeline be maintained.

At the beginning of the development, the scrum master set the tone by creating a team charter and organizing sprint planning. Teamwork is an important aspect of agile development so creating a team charter to laying down the ground rules for how the team is to interact and work together was absolutely vital. Sprint planning made it possible to estimate the amount of work and time each task would take and create a plan for the sprint. Both the team charter and sprint planning established the sprint expectations in one way or another. Throughout development the scrum master ensured that the daily scrums were productive and did their best to mitigate dependencies.

In the earlier stages of development, the developers worked together to create the scrum board. They took the requirements gathered by the product owner, broke them down in to tasks, assigned story points to them, prioritized them and placed the items in the product backlog. Throughout development, the developers maintained the product backlog so that the team knew what items were completed, in-progress, and still needed to be started. This allowed the team to keep track of their progress.

The testers used the requirements gathered by the product owner to write test cases to test the requirements. As developers finished code, testers made sure that the completed code met the requirements. They also communicated with the team throughout development to ensure the team understood the requirements of the project. When there was a change in requirements, the testers adjusted their test cases to meet the new requirements. The testers played a huge role in making sure the project met the given requirements.

User stories help the development team understand the requirements and break them down into small digestible tasks that are then organized by their size and how high of a priority the task is. This helps the team decide the order items should be taken care of and create an accurate estimate for what can get done in a sprint. Having a plan and an accurate estimate ensures that true progress is made. Working code is developed sooner when projects are developed in chunks. It is better to have half of the tasks completed into working code than all the tasks half done with no working code. Agile practices like team swarming make it possible for the team to work through the product backlog quickly and efficiently. As mentioned above, the developers update the product back log as the go along which also contributes to moving toward the definition of done because it helps the team keep track of where they are at with different items.

Part way through development, there was a change in requirements; the client wanted their booking site to have a focus on detox and wellness rather than just the most popular vacation destinations. Since we used an agile development method, we had room to adjust with the requirement changes. Even though there was a change in requirements and additional work, the timeframe remained the same. Testers addressed the change in requirements by adjusting their test cases while developers broke down the new requirements into tasks and added them to the backlog. Adjustments were then made to the existing code to meet the new requirements. Agile development allows for development to return to earlier phases which makes it possible to adjust to requirement changes like this. With code being developed in chunks, new requirements typically do not lead to as much wasted work as there would be if the project was not being broken down into chunks.

During development, there was a change in requirements, and it was not completely clear what the new requirements were. To remedy the problem, I sent an email to the tester to clarify:

Email to the test:

Hello (inset tester name),

With the recent changes in requirements, I wanted to ask a few clarifying questions regarding the specifics of the new requirements:

Does they client want vacations to be organized by popularity still?

Should there be a variety of types of vacations within the detox and wellness category, or will it be ok if multiple vacations with similar activities show up on the list?

Do we need to include a filter so clients will only be shown the kind of vacations they enjoy within the detox and wellness category?

Thank you,

Annaliese Pintar

The most effective part of my communication here is that I kept it simple and short. I didn’t clutter my email the unnecessary information and was direct with what I needed. This increased the chances that I would get the answer I needed sooner which means I would spend less time communication and more time making progress.

Some of the organizational tools we used that helped us be successful were user stories, product backlog, and the team charter. The team charter helped us be able to function properly as a team so that development could go smoothly with less conflict. The user stories helped the team understand the requirements and create the items that were put on the product backlog. The product backlog allowed the team to track our progress which helped us reach the definition of done. The organization tools made it possible for the team to function in an agile way and should continue to be implemented in the future.

Agile was the ideal development methodology for this project. The timeline for this project was short so the project needed to be developed rapidly. Agile is great for rapid and efficient development when done correctly. Also, the product has room for growth and change in the future, which the agile approach will allow for. There was also a change in requirements during development which we may not have been able to accommodate for if we did not take an agile approach. The only con is that with the ability to accommodate for requirement changes and the expectation that you are going to deliver within the timeframe regardless, there is a possibility that you will not get all the features you initially planned on where as with waterfall, you are delivering a completed product in the end.