There are four main roles that were used with this team for SNHU Travel, the Scrum Master, the Product Owner, Developers and Testers. Many times, there are other roles that are used in Scrum, such as dividing the testers into SIT and UAT, or dividing the developers into specialties, such as AWS, or UI however, our team did not require that level of specialization as the size was prohibitive as well as the knowledge of our testers and developers.

The Scrum Master served as a ringmaster, organizing, and prompting each of the other team members. The Scrum Master ensured that the team had everything they needed to complete their duties and that the team worked as a team, including things like encouraging pair programming for a particularly tricky bit of code.

The Developers of course were essential in developing the code for the new system. They were responsible for working together and in a couple of instances, quite literally by means of pair programming, where one person types, the other person observes and together they write the code. This helps with tricky code where two minds work together and reduces errors by having two pairs of eyes watching the code as it is written. It also ensures that the developers are cross trained in programming, that way there are no holdups. If a person needs to take time off there is another who can pick up where the other left off and work can continue uninterrupted.

The Testers work together to interpret the acceptance criteria as laid out by the Product Owner. They set up scenarios for test cases and run continuous testing. The testing is started early and frequently, and often repeated. As new changes go in, the testers test the previous cases to make sure the new changes did not impact existing functionality. This allows the code to be checked at many stages to identify issues as they arise rather than trying to identify the issues in a large block of code. They also worked closely with the Product Owner to answer questions and gain clarify in the expectations, in once instance taking back to the developers to correct a misconception on the way the display was done, as a list rather than a slide show of travel destinations, and if the destinations should display from 10th most popular to 1st, or 1st to 10th.

The Product Owner works directly with the stakeholders to obtain the requirements. They share the vision of the stakeholders with the developers, creating the user stories for developers and testers to follow and help to organize by priority in the backlog. The developers do have a say in how the stories are worked by discussing with the Product Owner each item and pointing out where there may be dependencies. For example, to create the list of top destinations, they need to have a means of tracking the popularity otherwise, they are just presenting destinations rather than Top Destinations. Together they determine that the tracking must be done before the high priority item of displaying Top Destinations. The testers also work closely with the Product Owner in making sure that they are sharing the same vision of the final product in testing. They also work with the Product Owner to determine if something is a defect or an expected result. The Product Owner is the face of business and is generally needed to be in constant communication with the entire team, responding promptly to questions and clarifications.

The Scrum-agile approach to the SDLC helped with the user stories because the team was able to break the work into smaller bits. Each team member was always active, and the stories being kept smaller allowed the team to see the results quicker. They were able to have a better sense of completion with each success. They worked together during the Daily Scrum calls to give reports and gain assistance with new perspectives on the issues they might be having.

The Scrum-agile approach really was vital when the Product Owner came to the team with a last-minute change. They wanted to change the vision of the Top Destinations from general popularity to a Health and Wellness focus to get ahead of an anticipated trend. We were able to take the change and modify a smaller portion of code without interrupting the overall flow of work. A single user made some changes with the testers adjusting some of their test cases. This supports the Agile approach of responding to change over following a plan (Blake, 2021). With the iterative work, and the work broken into smaller projects, the impact of the change was minimal, affecting only a couple of stories rather than throwing the whole plan into chaos.

Communication was kept extremely effective through multiple channels. The Daily Scrum gave everyone a chance to talk and present their work and barriers. They also communicated by email and text. One of the testers communicated with the Product Owner to get clarification on a couple of the user stories through email while one of the developers reached out during a meeting to ask for clarifications. The value of Individuals and interactions played out well because they were able to talk more effectively (Blake, 2021). Face-to-face communication helped to cut down on misunderstandings. The Grooming calls ensured the team had a chance to talk to the Product Owner and cover all the points needed to gain clarity while email and texts allowed for rapid communication between meetings. When change was required, the Product Owner was able to explain what was needed, and just as importantly, what was not needed in the change as well as making it clear that the timeline has not changed.

There are several ‘ceremonies’ within the Scrum framework that really helped the team to be successful. The Daily Scrum was especially effective in keeping the team moving in a forward direction and preventing bottlenecking and allowed the team to boast about their accomplishments. They enjoyed working together to overcome barriers and brainstorming at times. During the Daily Scrum, we began to employ Jira, which was quite effective. It allowed everyone a visual representation of their progress, showing who was working each story and watching the progress of the stories and giving real-time changes in ownership, such as when the story moved from development to testing or from testing to Product Owner for final approval. It also showed what was in the backlog and what could be moved to active work when they finished the current activities.

A velocity burndown chart was employed to show how much work the team was able to complete in a sprint and this was great because as the team became more unified and worked together better, they could see how their velocity increased.

The Scrum-agile approach was effective in many ways, not the least being the way it allowed the project to shift directions rapidly and with minimal impact to the work already complete. The team found that as they got more comfortable with each other, they were able to work on projects together and cover for each other as needed.

The cons included a certain level of uncertainty from the team, working directly with the stakeholders was very new and from a team unaccustomed to answering directly to the business, it was a bit stressful at first. There was also a lack of long term planning capable which for many people is not something they are comfortable with.

It is my opinion that the Scrum-agile approach is better than the waterfall method in particular because of the changing needs of the customer. As society changes, vacation destination desires change. Also, the requirements that people have from the system they work with are rapidly changing and Agile works with the rapid changes. Waterfall would not have been able to adapt as easily to the change in Top Destinations and would have required starting from the top again. Waterfall may be appropriate for projects that do not have rapid changes from business needs such as what SNHU Travel faces, but we need the ability to change quickly with the public demands.

References

Blake, S. (2021, February 11). Your Guide To Agile Software Development Life Cycles | Easy Agile. *Easy Agile*. https://www.easyagile.com/blog/agile-software-development-life-cycle/