Trent Hesler

CS250-T5475

6/18/2023

**Scrum Retrospective**

At the end of any Agile project, development teams look over the progress they’ve made and the steps they took to get there. Now that our SNHU Travel project has come to a close, I have been tasked with creating a Scrum retrospective. This retrospective is an exercise designed to encourage team members to assess the progress made and lessons learned during the development process. The idea is that, by looking back, we can have a clearer idea of what works and what doesn’t, moving forward. This allows an Agile development team to understand what worked and what didn’t, what went well and what could’ve gone better, allowing the team to maintain and improve the quality of their work, looking to the future.

As the Product Owner of the SNHU Travel Agile development team, my job was to create a vision for the product and to work with the client to determine completion criteria and user stories. To do this, I first sat down with stakeholders from SNHU Travel to discuss the ideas and requirements they had for the final product, an updated travel booking website. Once I felt sufficiently informed, I took the information I had gathered from these client meetings and created an initial vision of the product to present to the Scrum Master and Development Team. This initial vision is broken down into user stories (descriptions of the things that the finished product must do) and organized onto a product backlog (a sort of “roadmap” and progress tracker for the product’s development). I showed this product vision to the development team during our Scrum planning session. This planning session is a meeting where the development team and I collaborate to determine any necessary adjustments to be made to the product backlog and to evaluate approximately how many of the subtasks can be completed during each sprint (iterative development cycle). Following this meeting, I took somewhat of a step back from the team and allowed them to self-organize, as is customary. From this point on in the development cycle, I mainly served as an advisor to the development team and a liaison to the client. For instance, during the development of the SNHU Travel website, the client decided that they would like to shift the focus, from a generalized travel website, to one focused on health and wellness retreats. It was my job to communicate this change to the development team, groom the product backlog to accommodate it, and bring any concerns or requests for clarification from the development team to SNHU Travel. Through bilateral communication, we were able to make the requested changes to our product, without affecting our development timetable. Keeping open lines of communication helped our development team to think “what can I do to accommodate this change” as opposed to becoming overwhelmed.

From the initial Scrum planning session, the Scrum Master took over the day-to-day. The Scrum Master is responsible for keeping things flowing on the development team. Her job is to make sure that all members of the team are focused and empowered to do their work. She began each day of the development cycle by leading a daily standup, a meeting where development team members discuss what they’re working on, what’s going well for them, and what could possibly be improved. The Scrum Master takes this input and works to ensure that team members have the resources and support they need to complete the tasks of their current sprint and that any impediments to their progress are addressed.

The Development Team does the actual “meat and potatoes” work of creating and programming the software. On our development team, we used pair programming to ensure quality code was consistently developed. This is a process where two developers work together on a module, one developer “drives” or operates the computer and types the code, and the other developer directs and assists the first developer from the sidelines. During the development of the module, the developers take turns performing each of the two roles, ensuring that different perspectives are adopted and team members are cross-trained. During each sprint, the development team worked to create iterative pieces of code, starting with the base program functionality and then incrementing to improve and expand upon it. As the development team worked, they would mark off completed modules from the project backlog on a progress board, a bulletin board in the central office area, to keep the rest of the team apprised.

During and after each of these sprints, the Development Team passed off each new iteration of the code to our Testing Team. The testing team is a group of developers tasked with testing the code to ensure it is functional and free from errors. This was accomplished through the use of both automated testing practices, as well as manual unit testing. The testing team would put each new iteration of the code through its paces and ensure that it met the “definition of done” set for that sprint and that it accomplished the requirements laid out in the user stories. If bugs were found, the testing team worked to assess why these issues arose. They would take this information along with any other feedback they had and pass it back to the developers so that they could understand where they stood. Having a testing team that understands what they’re doing and is diligent and consistent in their testing practices is essential for delivering a polished final product.

As you have seen, each of the roles of an Agile development team have their respective roles to play and responsibilities. It is this diversification and interconnectedness that allows Agile development teams to be successful. No one person or team bears all the responsibility or pressure. This, in combination with the flexible, iterative development process, make the Agile development methodology an ideal strategy for projects that involve or can tolerate a higher level of uncertainty. Agile is structured to avoid rework and centralized points of failure, problems that plagued many waterfall style development projects. If a project, by its nature, involves a high level of uncertainty—or at the very least, does not require a high degree of upfront certainty—then Agile should definitely be considered.

However, there are, of course, scenarios where Agile would not be a good fit. For instance, if the project requires budgets, timelines, and other development criteria to be set in stone, then the flexible nature of Agile might actually come as a hindrance to the development lifecycle. While Agile still strives to meet solid timelines and budgets, it requires a degree of ambiguity that can be detractor.

In our case, Agile was a highly-effective and enjoyable methodology for the development lifecycle. It allowed us to incorporate changing guidance and diversify our developers skill sets. I look forward to bringing Agile forward with me to future projects and I’m thrilled at how well it worked for us during our work with SNHU Travel!