Tyler Hollingworth

CS-250: Software Development Lifecycle

Professor Michael J. Susalla

4-17-2022

**Sprint Retrospective and Review**

Within the Scrum framework, there are several key roles, responsibilities, artifacts, and ceremonies contributing to the success of the team and the product development lifecycle. The key roles within the scrum agile environment consist of the scrum master, product owner, developer, and tester that all work cohesively as a self-managing team. All these roles are equal in ranking with different tasks assigned to them for the completion of the product. There are also a few key artifacts that are essential to the scrum framework which include the product backlog, sprint backlog, and increment. Finally, several ceremonies are necessary to implement the scrum environment such as the sprint, sprint planning, daily stand-up/scrum, sprint review, and sprint retrospective.

Throughout the project, I played all the roles, worked with all the artifacts, and participated in all the ceremonies to meet the expectations of the client to create the Driver Pass system. The first role I took on was the scrum master and its responsibilities. In this role, I was responsible for ensuring that Scrum practices are known, understood, and fully implemented by the scrum team. This means that I facilitated scrum ceremonies, participated in creating and maintaining the product backlog, helped the team clear impediments, and developed the team to embrace a self-managing atmosphere within the scrum framework. This role is known as a servant-leader because the scrum master practices leadership by serving the team such as ensuring that the team has the correct tools/resources at their disposal and that they understand Scrum theory, practices, rules, and values.

The next role I took on in the project was the product owner role and all its responsibilities. The product owner acts as the ambassador to the project by being the central person responsible for communication between all the parties involved in the project such as the stakeholders, business organization, and team. As the product owner, I worked with clients to gather requirements for the system and refine them into user stories that would form the product backlog. I then brought that information back to the team so that the individual user stories could be prioritized and given story points that define their measure of effort required to reach a state of done. As the project progressed, I gathered greater detail on the requirements and embraced a just-in-time philosophy. I also groomed the product backlog as the user stories and priorities changed to match the business environment. Lastly, I communicated with the team on updates and kept them informed of the product vision at every interval.

The next role I took on was the tester role along with that roles’ responsibilities. As a tester, it was my responsibility to work closely with the rest of the team and develop test cases that align with the product vision. In the agile environment, these test cases are developed concurrently with the development team so that problems can be found early, fixed early, and save the project from a major rework later. To develop test cases that adhere to the product vision, I found that the “so that I can” portions of the user story were most beneficial in understanding what the requirement entailed and what needed to be completed to reach a state of done. As a tester, I needed to understand all the aspects of the user story and product vision so when any of these aspects were not clear to me, I would reach out and get clarification. The main method I used to do this was through email by asking probing questions and prompting for a response on the issues that I was not certain of.

The last role that I took on within the project was the developer role and responsibilities. As a developer, it was my responsibility to design and build the deliverable product within sprint iterations of 2 to 4 weeks. Much like the tester, I needed an accurate understanding of the user stories and product vision, so communication was an important aspect to progress in the development lifecycle. As the developer, I worked with the team to determine the user story priorities and story points so that the team would understand what requirements had the highest priority and if they could be accomplished in the upcoming sprint iterations. Lastly, I worked closely with the testers to understand test cases that needed to be passed so that the user story could reach a state of done.

Using the scrum-agile approach in the DriverPass systems’ software development lifecycle had many benefits for the team. The main benefit to the team was that it allowed for change in the project throughout its development which resulted in competitiveness in a changing business environment. Through constant communication between stakeholders and the product owner, the team was able to iteratively assess the product vision and groom the product backlog. The sprint review was the main touch base for this as all the involved parties came together to reflect on the product and make changes to better fit the vision of the clients and stakeholders. Ultimately this ability to be flexible and incorporate change in a business environment resulted in a better deliverable product to the client.

Using the scrum-agile approach removed the complete ownership from the project manager and placed it on the entire team resulting in an overall sense of empowerment from the team. The team also worked cohesively in many aspects, one of those ways was using scrum ceremonies. The daily stand-ups allowed for the team to participate in a brief daily meeting that addressed what was accomplished the day before, what they would accomplish today, and what was impeding their work. The sprint planning was a meeting where the team planned the upcoming iteration such as what user stories were the highest priority needing to be completed and based on story points which user stories could be incorporated into the upcoming iteration. The sprint retrospective was a meeting held at the end of the sprint iteration to discuss what processes worked and which ones hindered development so that the team could adjust to improve performance with every sprint iteration. The last ceremony which we already touched based on earlier is the sprint review which allows for all involved parties to come together at the end of the sprint iteration to discuss the product and user stories that have reached a state of done.

Communication is a necessary component of working on an agile team especially using the scrum framework. The idea of a communication radiator is one form of communication that gives the team a high degree of success. This method can be a physical representation such as a whiteboard where user stories are written on sticky notes and moved around the board to satisfy different goals. One goal can be to separate the user stories into groupings of similar story points, and another can be used to move the sticky notes into different buckets of ready, in-progress, and done. However, these communication radiators take on a digital format in modern times with the increase of distributed teams. This is accomplished by the team and organization using project management tools.

Project management tools such as Jira, Azure boards, and many others exist not only to create digital communication radiators but offer many other tools to assist the project. Azure boards for example give the team the ability to manage the product backlog, manage the sprint backlog, and develop team communication channels. These tools are simple to use and encompass all the aspects that an agile team will need minus face-to-face communication which can be supplemented in the form of video conferencing. These tools support simple creation, deletion, and managing options for user stories and even allow the user to drag and drop user stories where they are needed which is comparable to moving sticky notes on a whiteboard. Another aspect I found very beneficial is that some of these tools allow for snippets of code to be included in team communication and others even allow for links to GitHub repositories holding the working code for the product.

It is important to remember that clarifying conversations must take place between the team members so that the best deliverable product can be created. As we stated earlier during the tester and developer stages, I learned that incorporating email usage into clarification techniques can be very beneficial in resolving unclear requirements and product vision. Below I have included a sample of what that communication may look like when using probing questions to get proper replies and clarification.

**Example Email:**

To: Christy

Subject: Clarification

Dear Christy,

After reviewing your user stories, I have started developing test cases to determine whether the product passes or fails the requirements. I need additional information so that I can use more specific and detailed criteria in defining those test cases. Would you be willing to answer the following questions for me?

User story one:

* Should the slide show be based on the highest user ratings or the closest relationship to past destinations?
* Should filters such as drop-down options be available to toggle criteria like vacation type?

User story two:

* Should there exist a text box for custom price range options or a drop-down box with predetermined price increments?
* Should the price range option list destinations in ascending/descending order or merely remove destinations that are out of the price range and keep the current ordering intact?

User story three:

* Should the slide show go in order of the biggest discounts or the lowest price?
* Should the destination discount show a percentage of the savings, so the user knows how good/bad a deal is?

Thank you,

Tyler

In conclusion, I believe the scrum-agile approach was appropriate for this project as the requirements and scope were not predetermined, and the project was expected to be in a state of change as business environments evolved. Had this not been the case such as all the requirements and scope being known upfront with little to no room for change then the waterfall method could have been used on the project. The pros of using the scrum-agile approach are that the team was well equipped to address and even expected constant change during the development lifecycle. The agile method was also highly beneficial due to the stakeholders having the ability to remain involved in the entirety of the project. Another pro to this method is that all the team members had ownership and therefore empowerment in the project. Lastly, the testers were involved concurrently in the project with developers so that bugs could be resolved before progress went so far that a rework would be required to fix the problem. The only con that I saw within the scrum-agile approach was the scope not being known beforehand so budget and time restraints needed to be addressed iteratively as they would constantly be fluctuating.