Software Development Lifecycle

There are multiple common roles found among agile teams, these are Product Owner, Scrum Master, Developer and Tester. The product owner has the important task of understanding the client needs and transforming it into user scenarios that can be understood by the rest of the team. There are many traits of a product owner needs when dealing with clients some of these are patience, communication skills, positive language to keep the clients trusting, persuasive skills, taking responsibility and being empathetic to the client. When it comes to transferring client’s needs back to the development team some useful traits are technological knowledge to know what’s possible, understanding requirements, understanding perspective of work and being able to communicate well with development team and then have faith in them to follow through. User stories are a great tool for the scrum team to use. It helps to translate what the client says into what needs to be built and what task this will accomplish. By putting it in story format it simplifies the requirements creating clear concise directions for the scrum team. User stories typically cover three main topics who, what and why. It is during interviews with clients that user stories are developed. What the users say is turned into a story to discover exactly what functionality will help meet their requirements or preferences. User stories are also grouped into separate priorities and by having the client meetings the product owner can best prioritize each functionality the client desires.

The scrum master is a “servant-leader”. The point not specifically said even though somewhat insinuated from other traits is that he trusts the members of the team and doesn’t micromanage them. By serving others needs the strongest leaders are built. Often trusting a team member to accomplish a certain goal will give them the extra motivation they need to deliver an outstanding product as opposed to just a good one. If the scrum master doesn’t trust the skills of individual team members, it will hinder the individual progress. The scrum master is the most knowledgeable of scrum practices and helps the team to keep and develop agile practices.

The developer typically contains special skills to be able to accomplish the project. User stories, priorities, and test cases are designed from other members of the team and it is up to the developer to prioritize and complete deliverables as they see fit. Of course this is a team effort but the developer has a free range to accomplish these tasks in the best way they see possible.

The tester is responsible for how done a user story is to deliver it to the client. This is accomplished by using test cases for each part of the program to determine if the product is considered done. Acceptance criteria of user stories are developed to know when this point is attained and it is typically best agile practice to stop at this point and only add enough functionality to accomplish acceptance criteria.

Scrum approaches contain certain events that aid in the creation of developing and completing user stories. One of these tools is sprint planning.  A sprint is an approximately 1 month or less and is planned out which user stories are going to be accomplished in that time. My goal as a scrum master in planning a sprint would be stating the goals of completion for that sprint. There are three main topics that I would discuss during sprint planning.

1. Why is the sprint valuable?
2. What can be done this sprint?
3. How will the chosen work get done?

As the scrum master it would be important for me to keep the meeting on topic and give me a chance to find potential areas that the team will need extra support. Another important event is the daily scrum. This daily meeting gives me and the team an opportunity to see how progress is going on the sprint goals (user stories to be completed) and to discuss what will be accomplished that specific day and what can possibly hinder accomplishing those goals. I would hold the scrum to be a 15 minute interval and would allow the developers to run the scrum.

A backlog is a fluid list of tasks on user stories that still need to be completed. Backlog refinement is the process of breaking down a task into smaller increments to a point where it can be accomplished during one sprint. At this point it is then ready to be added in the sprint planning session. Often the tasks at chada tech such as the SNHU travel project can seem daunting however by breaking down each project into smaller projects the team can accomplish tasks and offering the clients deliverables. The last event is the sprint retrospective which is a look back at the previous sprint and to rehash issues that come often with people interactions or better development of the team. When flaws are found ways of improving are discussed to help improve the next sprint.

By dividing work into small chunks that can be accomplished in a short amount of time the project seems accomplishable as well it is always moving forward. Looking at the big picture is too daunting and inefficient work would be accomplished. This process also really helps teams become the best they can be by empowering them to work strong and hard together and reviewing where things went wrong it pushes everyone to be their best.

During the creation of the SNHU travel program there was a major design change by the client in the middle of development. Instead of the list view that the project began with the client asked to switch to a slide show view. This frustrated our development team at first. However as we quickly saw it only required some reworking of the code and the major functionality was not lost. The client knew that this added an extra spring cycle and they were ok with adding it because they received the functionality that would help them maximize sales.

Communication is the pinnacle of a great team. The most effective communication practice is the in person daily scrum meeting (stand up). It gives each member the opportunity to discuss what they finished yesterday and what they plan to finish today. This holds each other accountable and helps to prevent fear in talking with teammates. This meeting is a chance for others to offer support in an area they may be better versed and a place for the person to say where they may be lacking on a particular subject. User stories are also kept somewhat vague regarding specifics of how they will be accomplished. These finer details are something that are discussed face to face which promote teamwork lifestyle. At this point in agile development however the in-person scrum meeting is less and less possible due to remote lifestyle so these meetings are often taking place through online platforms which create their own difficulties in building openness and transparency within teams, but it is a viable solution.

Information radiator is also an important tool that allows teams access to a board (virtual or a literal board) that shows the teams progress, backlog etc. This can be seen by team members, product owner and often the client so that it is very transparent the work that is getting done.

During the creating of the SNHU travel assignment one tool that was integral was the use of GitHub in the development process. This allowed for all the remote developers to submit their work safely and if there ever became an issue with the project not working it became easy to see where the problem arrived and attempt to remedy the issue. This platform also allowed each team member to see when things were completed and when they were accepted as “done”. Our team used Microsoft for most tasks. This allowed easy file sharing and the use of Microsoft teams to hold the daily scrum meetings.

Using a program like JIRA helps to build a strong team. The app allows issue tracking and stories so that teams always know what is going on, this is essentially an information radiator. One of the features of JIRA that is useful is the ability to prioritize stories in order of importance. This way the team will begin working on the projects that need to be done first. The importance of the stories can be looked at from multiple angles such as customer value, implementation effort, operational cost, and risk.

Communication with the team is important an example email communication below shows how each member of the team should be working with other members to gain enough information to accomplish their task.

To: <Scrum Master><Product Owner>

From: <Product Tester>

Subject : Missing Information for SNHU Travel

Hey Everyone,

Great job so far on this project I am very excited to see how it progresses.

As I was building out the test cases there was multiple question, I had that were not available to me on the story boards. First Is it possible to prioritize all the user stories for me so that I can properly prioritize the stories for testing purposes?

Can we also talk more deeply about the change from standard list from the client to a slide show format? This greatly affects the test cases and updated user stories will help me better complete these tasks.

Thank you for you time,

<Product Tester>

The two main approaches to software development that could have been used for SNHU travel are waterfall and agile. For this project the team chose agile and this was a great choice especially due to the client changes in the middle of development that didn’t affect the project in the agile environment but would of significantly affected the waterfall model.

The waterfall model has the below attributes:

1. Waterfall was the first process to be introduced
2. It is very simple to understand and use
3. Each phase must be completed before the next phase
4. Waterfall model is a sequential design process in which progress is seen as flowing steadily downwards.

The advantages of the waterfall style are:

1. Departmentalization and Control
2. Easily Understandable
3. Each phase has specific deliverable & review process
4. Phases will be completed one at a time (There is no overlap)
5. This works for small projects when requirements are well understood

The disadvantages of waterfall style are:

1. It is difficult to estimate time and cost for each phase of the development process
2. Changing requirements will be completely re-work in project
3. Not a good model for complex and big projects
4. Not suitable for the projects where requirements are not fixed
5. The project will be delivered at the end