# Final Project Submission

Andrew Hager

Southern New Hampshire University

***Contributions from the various roles***

The various roles throughout our scrum team had all contributed in different and meaningful ways. The product owner had facilitated meetings between the users, stakeholders, and scrum master to pinpoint the different problems that the potential solution would aim to solve. By facilitating this meeting, the requirements from the stakeholders were laid out as to what they had wished to accomplish – a new set of software that would allow them to gain market share in a niche but up and coming market. After holding the meetings to gather the requirements, the product owner had then facilitated meetings with the scrum master and the users to understand the general concepts of how the users would expect the software to work and interact with them. From there, the product owner was able to define a set of requirements and features that could be translated into user stories that the scrum master could then put forth for the rest of the agile team to begin holding informational meetings (pre-scrums, daily scrums, and so forth) on.

The scrum master would take the user stories and the prioritized backlog that had been developed by the product owner and begin holding scrum meetings for the development team to streamline the process and get everyone working in sync. The scrum master had helped by assisting in leading the scrum and guiding conversation to identify anything that would help produce quality product, as well as find out where any problems and issues are arising. The scrum master had also helped to facilitate any last-minute changes that had to be done and prioritized. One key area where this had happened is when the scope of the project was changed last minute from niche vacations into health and wellness vacations. Without the effort of both the product owner and scrum master, this may not have gone over as well.

The development team had been instrumental by doing rapid iterations through their sprints, as well as helping to facilitate testing. Testing is a big area in scrum, through test driven development, and the development team had helped in any way they could through key communication with the testers. Documentation was supplied by the development team on the expected behavior and expected results for specific areas and user stories. By communicating through these documents with the testers, areas were able to be identified that needed work or had needed re-examined and potentially broken down into smaller and more specific user stories.

The testers are potentially one of the most crucial components of the team. They had documented tests and given suggestions as to what could be changed to better address the user story laid forth. One key area of this is when a user story had been encountered that used user profiles to highlight preferred vacation types. The testers were able to determine faults within the current set up and how the results would not be properly supplied due to the vagueness of the requirements and had offered suggestions to the development team to better address the issue going forward.

***Scrum and the SDLC***

The scrum-agile approach helped in every measurable way with addressing user stories throughout the software development life cycle. One key example that had stuck out was the change over in scope, addressing what type of software would be created. The software’s original scope was to supply niche vacations to the users for filtering, selecting, and purchase. Using agile, the requirements were able to be defined rapidly as information was gained on what would be addressed. When the change of scope happened, user stories were able to be recycled and used as is, with only certain key features being changed to address more of a content scope than a software scope. The switch was very rapid, tested quickly, and delivered with good results. If this would have been the waterfall method, numerous documents would have had to be changed along with an entire process of rewriting software – if it could have come to that. Agile allowed rapid adaptation through these user stories, and a very rapid change over to the new requirements will full testing performed.

***Interruption and Direction Change***

When the project had shifted focus, the scrum-agile approach allowed a rapid shift into what was immediately important and what was not a necessity. By focusing on what needed changed and what should not be touched, new user stories could be developed that superseded the predecessor user stories and allowed for a rapid shift in direction. The new user stories focused on the content type that would be addressed, replacing the old content. No new shifts were performed in how the software operated or what the software would perform. This allowed tests to be recycles, most user stories to be reused, and the framework to be left intact for the most part. Without scrum-agile, this could have been a lot more difficult. There would have been a lot more planning involved, a lot of unnecessary code rewrites, and little area for testing that could have degraded the quality of the product. Agile allowed the project to stay on schedule by addressing only what needed to be addressed. Due to how planning is performed in Agile, there are very few (if any) unnecessary meetings and replanning events. Sprinting commences as planned with very minor shifts in how the work is performed.

***Scrum Communications***

One of the biggest takeaways from communicating with my team was through the daily scrum. I will give examples below of what I had communicated, and how they were effective.

**Example #1**: I have noticed that when pair-programming is used, code quality goes up and we are able to catch more errors than if we are using isolated programming. We should utilize pair-programming whenever we are working on any critical components.

**Why it is effective:** This is effective communication as it gives the topic being discussed outright – pair-programming. Right after reaching the topic, I’m giving a why in reference to the topic. To finish, I give a suggestion on the topic of pair programming. By communicating using this method, a full definition of why I am discussing this topic is laid out in full, as well as how I feel we should approach this topic going forward. In this case, I feel that pair programming is important to many vital parts of the software as it allows many semantic errors to be caught outright, due to having a second developer present and working with the other. Semantic errors are very difficult, if not impossible, for tools such as an IDE to catch. Having another developer working through the code will help prevent these situations from happening, especially in critical situations. This will keep code quality at a high level, while reducing rework and code refactoring due to these very problems.

**Example #2:** I had encountered an issue with user story #7. The user story is attempting to address two points in one definition – filtering vacation type by profile preference and price. I feel this needs looked at and potentially split into multiple stories.

**Why it is effective:** I had been direct on which user story I had an issue with, and what exact issues I had. This allows the scrum master to document these issues, and what pertains to them. Afterwards, the scrum master can take this issue to the product owner for further investigation and reflection. If the product owner feels that the issue warrants a rework, then they can split the story into multiple stories, and rework the product backlog based on importance. By communicating in this manner, I give a what, a why, and a potential fix. This gives the scrum master more to work with and a larger area of understanding with the issue, rather than a simple ‘I have a problem’.

***Organization tools and scrum principles***

Although there are numerous tools and principles at the disposal of the scrum team, I feel the most impactful were transparency, collaboration, and iterative development. With the entire scrum team being transparent with one another, this allows a no-surprises approach to development. We can see what the product owner has in store, what they expect, and what is a priority compared to what is a nicety. Knowing these, we can approach the product development with an understanding of what issues we are trying to solve. This also prevented our team members from being sandboxed from one another. Developers knew what we were all working on and could assist each other. We knew what the testers were working on and expecting. This transparency lead into collaboration, giving us different ways to work with each other such as defining tests. Being able to define tests based off user stories through collaboration, there were no surprises for either end of our team. The testers knew what we expected, and vice versa. This allowed a more fluid and higher paced team environment. In one specific example, it allowed the testers to identify a problem area where the code solution to a user story was too broad and would need either re-evaluated by the scrum master and product owner or would need to implement a new way of handling the solution, such as using tags for better filtering. Iterative development allowed us to approach problems in short chunks of work, making us more adaptable. By approaching a set of user stories through each sprint, we were able to keep code clean and focused. This clean and focused code reduced the demand and resources of both the testers and development team and made the code more adaptable. This went full circle when the scope of our work was slightly changed by redefining our problem area. By doing everything in short iterations, adapting the new user stories to the current process was very simple and had kept us from scope-creep and feature creep by implying stopping points and allowing us to integrate this new scope into our next sprint.

***Effectiveness of scrum***

The pros of the scrum-agile process are that it allowed an effective and immediate result throughout our work by supplying user stories that needed addressed and what was expected throughout each user story. The user story supplied the who, the how, and what they wanted to accomplish. This was nice in that user stories did not define how the work was to be done, giving flexibility to the development team on how best to accomplish the goals. Developing in iterations also allowed for a lot of collaboration as it gave time through the various scrum meetings to discuss what was working, what was giving us issues, and what solutions we may have.

The cons of the scrum-agile process seem to be getting a project up to speed. By leaving much of the decision making up to the development team, we have to figure out what tools and technology stacks are best, what are the best approaches to solving the problems, and so on. Even writing unit tests may be a con as bias could creep into these tests, requiring full collaboration between the different team members such as the testers and developers. Another big con that I have seen within scrum is being high performing. This is an easy thing to disrupt. If one team member leaves or is replaced, this could bring our team from a high performing team back to square one and building back up to high performing again.

Was the scrum-agile approach the best approach for our SNHU project? Yes. The agile approach allowed a lot of iteration throughout the different modules that may have not been possible with a monolithic approach such as the waterfall model. The biggest key takeaway that I will use is the change of scope. With the waterfall model, changes such as this could have disrupted the entire process and lead to a lot of rework and more testing, before being sent back for fixes. Utilizing the agile approach, the scope change was just another day of sprinting. Being able to just fall back in, iterate at what needs changed and nothing more, and then move forward to the next set of user stories is something that I feel only the different agile approaches would have allowed due to their flexibility and iterative-friendly approach.