Cody Belanger

CS250

10/11/23

Final Retrospective: Module 6 Final Project

Throughout this course and the Agile environment it emulated, I feel I learned a lot and got real good work done during. As the Scrum Master, I will be writing a retrospective on the events of this project, SNHU Travel, and what I learned in the various roles. There was a lot to learn and many challenges to overcome, so I hope to address that all in this final paper, allowing me to reflect on what was good, and 3what was bad in an Agile environment.

All of the roles were very interesting, and fun to try out. I think the diverse group of many *types* of talent in one team really helps foster a good Agile environment, allowing things to be done on the fly and changes made quickly. As the developer, being able to hear that the Product Owner talked to the client and needs changes, and get started on those changes within minutes was a great example of how clear communication lines and quick means of relaying information is key to a good Agile environment.

User stories is a great development methodology that I was introduced to in this class, and seeing big ideas broken down into bite size, infinitely more managable ideas, makes for a great way to make progress quickly. During my time as the Scrum Master, taking larger ideas and breaking them up into their most important components feels natural and how problem

solving in SWE *should* be, as it can feel daunting to hear the product owner say that they need a personalized timeline of travel destinations, but if you break that down into a picture viewer program, with text accompanying each image and an option to book the dates, it all becomes much more do-able, it feels much more realistic to give those multiple stories to various developers, than to attempt to assign 3 developers to work on the website at once.

When a project has a wrench thrown into it, and things change, a lot of things need to be done *quickly*. Agile is great for this as it allows to quick pivots with minimal downtime, when a client changes their mind or decides something isnt good enough and we need to re-work it, Agile allows the Product Owner to inform the Scrum Master and the entire team ASAP, so that no one is left in the dark. Quickly getting new information onto the vision board and all team members briefed on whats coming next, and what need to change, is easy to get done as soon as you hang up the phone or read the email with the new information.

The most helpful tools in a Scrum-agile environment is the vision board or whatever you use as an information radiator; this allows your entire team to be up to date about everything, even if its not affecting their role directly, it gives them a better understanding of all parts so that they know what contexts they are working within. Clear communication is key to keeping things consistent and not letting errors crop up undetected.

I think Agile is the best way for something like SNHU travel to be developed, especially in hindsight when we see how a not-so-minor but critical change happened part way through. For a project like SNHU Travel, Agile allows for the most effective quick-communication to keep people doing simultaneous tasks without interrupting flow. Allowing the developers to send the up-to-date code to testers so they may troubleshoot and ensure no features lack critical functionality, all while the developers continue to finish more user stories. This level of

overlapping work allows things to be done *much* faster than in a waterfall environment; where testing would only come *after* major portions of code (if not most of the program) has already been created, making bug catches more cumbersome than in an Agile environment, as even small fixes require major revisions late into program development.

As I also said, when the client makes not-so-critical changes partway through the development process, it definitely causes some extra work, and might even extend deadlines, but compared to a waterfall environment where that kind of change would require much more work done as the program would be nearly or completely finished at that point. By the client being able to reach out and express their desires during a sprint, and the team being told at that stage of development, things could be done much faster and efficiently. This is a major bonus to Agile as clients in a waterfall system may overload the Project Manager with too much, hoping to get as much as possible, but this can lead to worse results.

Overall, as I said I think that Agile works best in our modern development environments like the one we simulated in SNHU Travel, and I think it works best in any environment where change is possible, or inevitable; it seems these days all things will be changed or altered part way through, but I'm sure there's many applications where waterfall would be better suited. In the situation and development cycle we wimulated in this class Agile definitely seemed to be a better approach. Having to re-write major portions of code because they wanted a different visual style would be a major setback, but due to the timing and Agile environment, it was easier to mitigate and account for by changing less because more parts were still in progress and able to be better modified.