Mark Holden

CS-250 Software Development Lifecycle

12/12/2021

Sprint Review and Retrospective

The various roles of my scrum-agile team each had their own specific contribution to the success of the SNHU Travel project. The scrum master role contributed by identifying and removing impediments from all other team members including developers, testers, and even the product owner. The product owner’s role contributed the vision for the team and enabled all other team members to make the right decision at each step without getting analysis paralysis. The testers were extremely important in translating the product vision into user stories that can be tested and verified to ensure that each subsequent release is free of regressions. The developer(s) contribution beyond the obvious development of the application is that they automate testing so that the testers are not bogged down in the menial task of manually testing each bit of functionality which frees them up to be more proactive in the creation of effective test plans. One example of testing being done manually that should be automated is that last week, I found out that one of the projects I am responsible for is being tested manually by senior developers on a set of known inputs across a defined set of environments. A few weeks ago, I created the basic integration test project that simply verifies the program runs, connects to the required databases, and can run the simplest of its functions. With the list of cases executed before each deployment, I can enhance the integration tests, so that they are not done manually.

A scrum-agile approach to the software development lifecycle helped to bring each of the user stories to completion by focusing the team’s effort on completing the requirements of the story. Because each user story has, or at least should have, a clear expectation, when the developer is done coding and testing, the user story should be ready for quality assurance and user acceptance testing. According to Martin, “we will not, as a matter of ethics, knowingly ship code that is defective or substandard” (2020). That means that once the developer is done writing the software, other than verification, it should be good for production. In my opinion, the job of the tester should be upfront and proactive by the creation of test cases that the software engineer can develop to and write integration tests for. If that is done correctly, then there isn’t the need for time in the schedule for a large amount of additional QA after development before release to account for sending all the work back to fix bugs. A couple of weeks ago, the team that I worked on attempted to conduct pair programming. I have not done pair programming before, so I believe that we have a lot of work to do to get good at it, but we were writing tests, coding, and conducting the code review all at the same time. If that were made into a regular practice it could improve overall code quality and the speed at which we develop software.

Project completion was supported by scrum-agile when the project was interrupted and changed direction by empowering the client to make large architectural changes at the eleventh hour. As a software engineer, I love to see the fruits of my labor being used in production especially when it had a big impact that helps people and makes their lives or jobs better or easier, so when something that I have worked hard on gets completely scrapped it does not feel great. In the SNHU Travel example, part of the script was that the product owner and the client would come up with a change after development has already started so that students could be told that agile is superior in handling changes. In my opinion, having developers work on one thing, and then throw away that work to change direction is a waste of the client’s money, and affects the morale of developers. I always understood the sprint to be an immutable, or as close to immutable as possible, being the unit of work that everyone agreed upon in sprint planning. To have the product owner pull the rug out from under the developers will make it harder for them to deliver something at the end of the sprint. If the team delivers a substandard product at the end of the sprint it may shake the confidence of the client. If there do need to be changes implemented, or the direction changed, that is just fine, and there are ways to make that happen in both waterfall and agile. In agile, the next sprint can be devoted to developing for the new requirements, but in the meantime, the team has delivered a product that had been agreed to in sprint planning, and they have a solid release in the client’s hands. The same thing is possible in a waterfall approach by updating the specification document and notifying all affected parties, thus, the product is modified to accommodate the requested changes.

Communicating effectively is an important skill for software engineers to have on any type of team. One example of a time where I needed to communicate effectively was when I was in the Marine Combatant Dive Supervisor course in Okinawa, Japan. Third Recon Battalion was putting a course on and as the Dive Supervisor I would be responsible for the dive side, and the safety of all personnel. One of the things that the instructor staff did not like about my communication style is that I do not like yelling at people. In the Marine Corps, there is the assumption that loud communication is effective communication. In order to test how I would communicate in a stressful scenario, during one of my practical application tests, the role players were instructed to be loud, goofing off, and not paying attention to my directions. The intent of the instructor was to see if I would raise my voice to maintain control of the side. I knew immediately what was going on and instead of raising my voice, which in my opinion would not be effective communication in that case, I sternly told the individuals goofing off that it was of paramount importance for them to focus on the job at hand and give me their attention. This resulted in a successful and safe dive and my certification as a Dive Supervisor. The equivalent of this in the context of software engineering is that the most effective communication is rooted in the fact that each member of the team truly wants to do the best job that they can for the team. When the team leader of a software team gives the team clear tasks and shields them from distractions, they can really get a lot done.

The organizational tools and scrum-agile principles that help the team that I am on to be successful are primarily the daily standup and user stories. We do a text standup every day where we fill in any impediments we are having and what we will be working on for the day along with a sprinkling of questions that are asked less often, such as for feedback on any process or technical improvements that can be made. In addition to the text standup, we join a Zoom meeting three times a week so we can talk through anything in the standup and get “face to face” with the other members of the team. We do not have the ability to do this in person because none of the members of the team live anywhere near each other. The other thing that really helps our team is the user stories. Our process requires a ticket to be scoped in detail before it is worked on, and in effect that scoping is equivalent to creating the user story. In addition to acceptance criteria, since all scoping of tickets is done by a software engineer, technical details are added to point the engineer that works on the ticket in the right direction. When I first started at this company this seemed like cheating because other places I had worked, that research was part of working on a ticket. Now that I am the one doing a lot of the scoping, I like it because it gives me a chance to think more strategically about how to implement the ticket rather than just hacking away until something works.

The agile approach did not seem to be very effective for the SNHU Travel project to me. The main issues that I saw were that there was a time where the team and the client were out of sync. This is not necessarily the fault of the agile approach, but in the example given, the product owner sent an email to the developer with almost a completely different product vision than was provided previously. Either I as the developer grossly misunderstood the requirements, or the requirements had a massive shift. One of the cons of an agile approach is that if there is no specification document that is the source of truth for what the project vision looks like, and the client thinks the product owner has the same vision when they do not, then the product owner communicates that incorrect vision to the team, where it can once again be misinterpreted, and by the time the first release happens the product looks nothing like what the client expected. One of the pros of the agile approach would be that there is a sprint retrospective, where concerns about playing the telephone game with client requirements can be addressed.

The intent of the class appears to be to sell agile training courses and books and convince students that their future employers need to hire agile coaches. In that sense, it was the best approach for the SNHU Travel development project. If the project was a real project, and the client was a real client, more information would be required to determine what the best approach would be. Based on other clients that I have seen that wanted contract development to be done, I would say the most effective solution would be some hybrid between waterfall, agile, and common sense. If the client has never heard of agile before, they may really enjoy the flexibility, but they may not understand the price of that flexibility in terms of what gets deployed to production. If the company is set up to provide a team to the client, and the client is happy to pay for that level of service, then pure agile might be the way to go. Many times, the client needs to have budget approval for expenditures, and not knowing how much a project is going to cost in advance can be a non-starter. One of the most challenging projects that I have ever managed was with General Electric because even though our stakeholder was great and understood the uncertainty of some aspects of software development, she had to answer to her finance department, who only saw the project as a never-ending capital expense. Attempting an agile approach on that project was attempted several times and could never get off the ground. In fairness to GE, there were issues on our side that were also hampering the implementation of agile that made it an uphill battle as well, so ultimately the team and client ended up with a workable hybrid, mostly waterfall, a little agile, and a little common sense.

References

Martin, R. C. (2020, June 22). Uncle Bob - Expecting Professionalism. CS-210 UncleBob Expecting Professionalism Kuppelsalen, Copenhagen (CC). Retrieved December 12, 2021, from https://youtu.be/KroXOP6qkxo?t=1439.