Sprint Review and Retrospective for the SNHU Travel Project

Roland Pratts
Southern New Hampshire University
CS 250: Software Development Life Cycle
Instructor: Professor Martann Krupa
Date: 12-20-2025

During this course, I worked on the SNHU Travel project while rotating through different Scrum-Agile roles: Scrum Master, Product Owner, Developer, and Tester. For the final sprint, I am acting as the Scrum Master, leading the Sprint Review and Retrospective. Even though I was not part of a live team, each activity in the course gave me practice with the kinds of work a real Scrum team would do. This paper explains how the different roles supported the project, how Agile helped user stories reach completion, how interruptions and changes were handled, how communication and tools supported collaboration, and why Scrum-Agile was a good fit for the SNHU Travel project.

1. Applying Roles on the Scrum-Agile Team

As Product Owner earlier in the course, my main focus was the product vision and the backlog. I practiced writing user stories that showed what travelers needed, such as being able to search for trips, review options, and book safely. I learned that a clear vision helps everyone move in the same direction. If the team knows who the users are and what problem we are solving, they can make better choices about what to build first. I also realized that empathy is an important trait. A good Product Owner listens to both customers and the team, and adjusts the backlog when work is unclear or too large.

As Scrum Master, my focus shifted to the process itself. I planned how to run Sprint Planning, Daily Scrums, Backlog Refinement, Sprint Reviews, and Sprint Retrospectives. In the Vision Quest Software proposal, I described how the Scrum Master protects the team from constant changes, removes blockers, and creates a safe space for honest feedback. That work helped me understand that the Scrum Master is not a “boss,” but a servant leader who supports the team and helps everyone follow Scrum in a healthy way.

As a Developer, my attention moved toward building working software. In assignments that acted like SNHU Travel features, I had to think about estimating work, breaking tasks into smaller pieces, and delivering code that actually runs. This matched the idea of small, cross-functional teams, where developers own a feature from design through to testing and demo. Being in this role showed me how important it is to have clear acceptance criteria and a realistic sprint commitment.

As a Tester, I focused on building quality into the product from the start. I practiced thinking about test cases based on user stories and acceptance criteria, instead of waiting until the end. In the Vision Quest plan, my tester role emphasized Test-Driven Development (TDD), automated tests, and continuous integration. That experience showed me that testers in an Agile environment are not a separate phase; they are part of the team's day-to-day work, giving fast feedback as soon as new code is written.

Working through all four roles helped me see how they fit together. The Product Owner sets direction, the Scrum Master guides the process, the Developers build the product, and the Testers help make sure it works as expected. When each role does its part and respects the others, the team can deliver more value each sprint.

2. Completing User Stories With Scrum-Agile

A Scrum-Agile approach helped user stories reach completion by breaking work into small, testable pieces and running all phases of the software development life cycle (SDLC) inside every sprint. Instead of doing requirements, design, implementation, and testing only once in a long project, we cycle through those steps for a handful of stories at a time.

For SNHU Travel, I treated each assignment or feature like a user story, such as "As a traveler, I want to see a clear list of trips so I can compare my options quickly." In a real sprint, the team would refine this story, add acceptance criteria, estimate its size, and then commit to it during Sprint Planning. During the sprint, developers would design and build the feature while testers prepare and run tests.

One powerful idea that supported completion was the Definition of Done. In the Vision Quest proposal, our team agreed that a story was done only when it met all acceptance criteria, passed manual and automated tests, was integrated into the main branch, and was accepted by the Product Owner. Applying that same mindset to SNHU Travel meant that "done" was not just "I wrote some code," but "this feature works, is tested, and is ready to show." This helped avoid the "illusion of progress," where work looks busy but nothing is truly finished.

3. Handling interruptions and changes

Real projects rarely go exactly as planned. Tools break, new requirements appear, and earlier assumptions turn out to be wrong. I saw smaller versions of this during the course. For example, the SNHU Travel ListView customization assignment turned out to be harder than expected because of IDE problems, jar file issues, and debugging time. What seemed like a “small” task ended up being closer to a “medium” or “large” task.

Scrum-Agile supports these kinds of interruptions by accepting change as normal instead of treating it as a failure. In Scrum, the Daily Scrum is where the team surfaces blockers. The Scrum Master can then help remove those blockers or adjust the plan. If a tool breaks, the team might decide to shift focus while the issue is fixed. If a stakeholder changes priorities, the Product Owner can reorder the backlog so the most valuable stories come next, while keeping the current sprint as stable as possible.

Agile estimation methods, such as T-Shirt Sizing, also helped me think more realistically about change and risk. Instead of pretending we know the exact number of hours, we compare tasks by size and complexity. In my reflection on the ListView task, I realized that giving it a simple “small” label hid the risk of environment problems. A better estimate would have treated it as a larger task with more unknowns. Agile gives teams room to learn from these mistakes and improve their estimates over time.

4. Communication and collaboration

Good communication was at the center of almost everything I did in this course. My discussion posts, case study reflections, and Vision Quest proposal were stand-ins for the conversations that would normally happen in a live Scrum team. These pieces helped me practice how to explain ideas clearly, connect them to Agile values, and respond to feedback.

For example, in my Product Owner reflection, I wrote about sharing and promoting the product vision. That kind of message would be useful in sprint planning or stakeholder meetings, because it reminds everyone why the work matters, not just what the tasks are. In the Valpak case study, I explained how shorter sprints and frequent communication helped the company respond faster to

changing business needs. I also described how I would add automated testing to catch issues early, which links directly to Agile's focus on continuous improvement.

My post on estimation, where I discussed T-Shirt Sizing and the poorly estimated SNHU Travel assignment, was another example of effective communication. It showed how a team can talk honestly about underestimation and risk without blaming individuals. Instead, they can adjust their process and become more accurate over time.

What made these communications effective was that they used simple language, stayed focused on shared goals, and invited collaboration. This matches Agile values that prefer individuals and interactions over heavy processes and tools. Clear, friendly communication makes it easier for team members to trust each other and work together toward a common outcome.

5. Organizational tools and Scrum-Agile principles

Several Scrum tools and events helped keep the project organized. The product backlog provided a single ordered list of work items. This made it easier to see which SNHU Travel features should come first, usually those with the highest value to travelers. Backlog Refinement helped break down large stories, clarify acceptance criteria, and prepare items for future sprints.

Sprint Planning helped the team set a realistic goal and choose which user stories to commit to. The Daily Scrum helped keep everyone aligned by answering three simple questions: what did I do yesterday, what will I do today, and what is blocking me? Sprint Reviews focused on showing working software to stakeholders and getting feedback, while Sprint Retrospectives gave the team a space to reflect on what went well, what did not, and what to change next time.

These events match key Agile principles, such as delivering working software frequently, using working outputs as the main measure of progress, and regularly reflecting on how to become more effective. By connecting my course work to these real Scrum events, I saw how process and tools can support, rather than replace, good teamwork.

6. Evaluating the Scrum-Agile approach

Based on my experience, the Scrum-Agile approach was a strong match for the SNHU Travel project.

The main advantages were flexibility, frequent feedback, early quality, and clear roles. Short sprints and regular reviews made it easier to adjust when we learned something new. Having testing and integration built into each sprint supported the idea of catching defects early instead of waiting until the end. The roles of Product Owner, Scrum Master, and Developers/Testers gave everyone a clear responsibility while still encouraging teamwork.

There were also some challenges. Scrum includes many events, and at first it can feel like too many meetings. Agile estimation and story points can be confusing until the team gains more experience. There is also a learning curve when introducing practices like TDD or continuous integration. However, these challenges mostly exist at the start. Over time, a team can tune its process to keep only the events and practices that add real value.

For SNHU Travel, where requirements can change and user experience is important, I believe Scrum-Agile was the best choice. It supports collaboration, early feedback, and continuous improvement, which are exactly what a modern travel application needs. This experience taught me how the Scrum roles and events work together and how an Agile mindset can turn a static plan into a living, adaptable process.

References:

Cohn, M. (2019). Agile estimating and planning. Pearson.

Schwaber, K., & Sutherland, J. (2020). The Scrum guide. Scrum.org.

Sliger, M., & Broderick, S. (2020). The software project manager's bridge to agility (2nd ed.). Addison-Wesley.

Agile Alliance. (2024). 12 principles behind the Agile Manifesto. https://agilemanifesto.org