Eric May

Southern New Hampshire University

CS 250 Software Development Lifecycle

Jason Richard

06/21/2025

The SNHU Travel project marked my team's transition from a traditional software development approach to an Agile methodology using the Scrum framework. As the Scrum Master during this final sprint, I facilitated a Sprint Review and Retrospective to evaluate our progress, reflect on team roles, and assess the effectiveness of Agile practices throughout the development. This reflection includes a comprehensive look at how different Scrum roles contributed to success, how user stories were completed, how interruptions were handled, how communication influenced collaboration, and how Agile tools enhanced the process.

Throughout the course and project, I assumed several roles within the Scrum-Agile team: Product Owner, Developer, Scrum Master, and Tester. Each role uniquely contributed to the project's success

As Product Owner, I prioritized the product backlog and defined the acceptance criteria for key features. For example, when developing the "Top Destinations Interface," I clarified that users need to filter by destination and budget, which helped the development team focus on high value features.

As a developer, I interpreted user stories into functional components. In one sprint, I worked on implementing the slideshow interface that showcased destination images. I broke the task into smaller steps: loading images, creating transitions, and enabling navigation. This role depended on my understanding of Agile's incremental development.

As the Scrum Master, I facilitated daily stand ups, sprint planning, and retrospectives. I helped the team stay on track and removed impediments such as unclear requirements or overlapping task ownership. For example, when one team member's test case conflicted with the design specs, I coordinated a discussion to resolve the inconsistency.

As a tester, I created and executed test cases aligned with the acceptance criteria. I validated functionality, usability, and responsiveness across devices. This role reinforced the importance of continuous testing in Agile to maintain quality without delaying delivery.

The Agile framework proved essential in helping user stories reach completion. Rather than planning the entire project upfront, we broke the work into sprints and user focused stories that we could deliver incrementally.

One clear example involved the development of a travel package filter system. The user story stated "As a traveler, I want to filter vacation packages by location and price so that I can easily find options within my budget." This was broken into design, development, and testing tasks. Because we had held sprint planning meetings and maintained a clear backlog, we were able to complete this feature in one sprint.

Frequent retrospectives also allowed us to refine user stories and clarify acceptance criteria before work began. In a traditional waterfall model, we would not have had the same opportunity to adjust requirements once the development phase started. Agile's flexibility helped ensure we delivered stories aligned with evolving expectations.

Agile's adaptive nature allowed us to navigate changes during the project effectively. One of the biggest changes occurred mid-sprint when new requirements were introduced to include promotional discounts for new customers. Instead of derailing our timeline, we added this as a new user story, assessed its priority, and scheduled it for the next sprint.

Another example was a shift in the visual design requirements. Originally, we planned to use stock photos for the slideshow interface. Midway through the sprint, the design team requested higher resolution destination imagery. We adjusted by updating the asset pipeline and splitting image optimization into a separate task. These adjustments demonstrated Agile's strength in responding to real-time feedback without significantly impacting progress.

Had this been a waterfall project, such changes would have required going back to the planning or design phase, delaying overall delivery. Agile allowed us to remain flexible and continue moving forward while integrating evolving needs.

Communication was a cornerstone of our Scrum-Agile implementation. Even though the course was asychronous, I practiced simulated stand-up updates and shared feedback with peers. One example of effective communication was during our sprint planning discussion for the slideshow feature. I presented a proposal outlining the visual layout and navigation controls, and another team member responded with suggestions to simplify the interface. Our back-and-forth resulted in a cleaner, more intuitive design.

I also shared user story clarifications and sprint progress via discussion posts, mirroring what real-world Agile teams might communicate in Jira or Microsoft Teams. These updates encouraged collaboration by keeping everyone informed and aligned. Our simulated retrospectives encouraged open feedback and fostered a continuous improvement mindset.

What made this communication effective was its consistency, clarity, and focus on solutions rather than problems. By referencing specific user stories and outcomes, the team remained focused on project goals and was empowered to make informed decisions.

Throughout the project, we used several organizational tools and Scrum events that supported success:

Product Backlog: Maintained a prioritized list of user stories.

Sprint Planning: Set clear goals and estimated workload for each sprint.

Daily Stand-Ups: Used discussions to simulate progress updates.

Sprint Reviews: Evaluated working features and gathered feedback.

Retrospectives: Identified what went well and what to improve.

While these tools were used conceptually rather than in real-time platforms, their impact was still evident. The backlog helped us avoid scope creep, while sprint planning ensured we didn't overcommit. The retrospective process in particular helped the team reflect on both individual and team performance, fostering accountability.

Scrum's principles-transparency, inspection, and adaptation-were reinforced through these practices. Each sprint brought an opportunity to deliver working features, gather input, and adjust based on results, all of which kept us focused and flexible.

The Scrum-Agile approach proved to be well suited for the SNHU Travel project. Its structure promoted short feedback loops, empowered collaboration, and allowed flexibility in scope management.

Pros:

- Adaptability to change during development
- Early delivery of working software
- Improved communication and team alignment
- Strong user-centered design through iterative feedback

Cons:

- Requires high engagement from all team members
- May lack clarity if roles or backlog items are not well defined
- Difficult to scale without experience and strong facilitation

Given the project's evolving scope and need for iterative refinement, Agile was clearly the best choice. A waterfall approach would have been rigid and less responsive to changing requirements or new user insights. While Agile does demand more team interaction and careful backlog management, the benefits far outweigh the drawbacks.

The SNHU Travel project served as a successful test case for applying the Scrum-Agile approach to the software development life cycle. By rotating through key Agile roles and participating in essential Scrum events, I gained practical experience managing user stories, handling change, and promoting collaboration. The Agile process enabled us to build a product incrementally, gather timely feedback, and adapt to new insights. Based on these outcomes, I would recommend that ChadaTech move forward with transitioning its development teams to Scrum-Agile.