Glorimar O’Conner Lopez  
Southern New Hampshire University  
CS-250: Software Development Lifecycle  
Professor Tenario Powell  
October 19, 2025

Throughout the SNHU Travel project, each Agile role played a distinct part in guiding the software development life cycle. As Scrum Master, I coordinated sprint planning sessions, daily stand-ups, and retrospectives. I facilitated task assignments and ensured that any impediments were quickly identified and addressed. As Product Owner, I prioritized the backlog based on evolving client needs, refined user stories for clarity, and engaged in close collaboration with the development team to align deliverables with business value. In the Developer role, I participated in implementing features, validating user stories through test cases, and iterating on feedback. One example of these roles in action occurred during the sprint to integrate search filters into the SNHU Travel app. The Product Owner clarified acceptance criteria, the Developer coded and unit-tested the feature, and I, as Scrum Master, helped remove delays by ensuring access to necessary testing environments. Each role complemented the others, demonstrating the importance of cross-functional collaboration and Agile role clarity.

Adopting a Scrum-Agile framework streamlined user story completion by emphasizing iterative development and continuous feedback. Stories were broken down into manageable pieces and assessed for complexity using planning poker. For example, during the sprint focused on user authentication, we initially underestimated the complexity of implementing secure login. A mid-sprint review prompted us to divide the story into two: one for front-end UI and another for backend validation. This flexibility, combined with the daily Scrum meetings, allowed us to adjust the backlog and maintain velocity without compromising quality. In contrast to waterfall methodologies that delay feedback until final delivery, Agile empowered our team to deliver working increments of software at the end of each sprint, improving stakeholder satisfaction and team responsiveness.

Midway through the project, a shift in stakeholder priorities introduced new constraints on the user interface design. This change could have derailed progress in a traditional SDLC model. However, because of Agile’s embrace of change, we were able to adapt quickly. The updated UI requirements were discussed in the next backlog refinement session and integrated into the next sprint. By leveraging Agile ceremonies such as the Sprint Review and Retrospective, we maintained open communication, adjusted scope, and kept the team aligned. Our ability to pivot while still meeting deadlines highlighted Agile’s value in handling unexpected interruptions. This situation echoed Cobb’s (2015) view that Agile is inherently empirical, encouraging teams to learn and improve through experience.

Clear communication was crucial to the team’s success. For instance, I sent the following message to the team during a sprint: "Hi team, just a quick reminder before tomorrow's stand-up: Please update the task board with your progress and bring up any blockers related to the booking feature. We want to ensure we’re ready for integration testing by Friday." This message was effective because it reminded the team of expectations, encouraged ownership, and set a collaborative tone. By maintaining daily check-ins, progress transparency, and respectful dialogue, we were able to resolve conflicts early and maintain a shared understanding of project goals. Communication was the thread that held our sprint efforts together and ensured forward momentum.

Trello and shared Google Docs were our primary organizational tools. Trello was used for managing the product backlog, tracking sprint progress, and visualizing workflow through boards and lists. Google Docs facilitated collaborative editing of meeting notes and test cases. These tools were directly tied to Scrum events such as sprint planning, daily stand-ups, sprint reviews, and retrospectives. According to Cobb (2015), Agile tools are most effective when used to reinforce a spirit of collaboration and trust, rather than simply for tracking tasks.

The Scrum-Agile process provided a flexible and adaptive framework for the SNHU Travel project. Among the pros were improved team collaboration, continuous delivery of working software, and the ability to incorporate changing requirements. It fostered a culture of ownership and rapid feedback cycles. However, some cons included initial uncertainty due to vague user stories, uneven team participation in sprints, and occasional misalignment on sprint goals. Despite these challenges, the Agile approach was ideal for this project. The iterative delivery model allowed us to build and refine features incrementally, which was essential given the evolving vision of SNHU Travel. As Cobb (2015) notes, a successful Agile implementation must go beyond mechanical rituals and embrace cultural and organizational change. Our experience mirrored this idea: when the team actively engaged in Agile principles, outcomes improved significantly.

The SNHU Travel project demonstrated the tangible benefits of Agile when implemented with discipline and collaboration. By rotating through Scrum roles, adapting to changing conditions, and using Agile tools effectively, our team delivered a functional and flexible application. While not without its hurdles, the Agile methodology proved to be the most effective approach for this dynamic development environment.

# References

Beck, K., & Andres, C. (2004). Extreme Programming explained: Embrace Change (2nd Edition). In *Addison-Wesley Professional eBooks*. https://dl.acm.org/citation.cfm?id=1076267

Cobb, C. G. (2015). *The Project Manager’s Guide to Mastering Agile: Principles and Practices for an Adaptive Approach*. John Wiley & Sons.

Highsmith, J. (2009). *Agile Project Management: Creating Innovative Products* (2nd ed.). Pearson Education.

Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P., & Still, J. (2008). The impact of agile practices on communication in software development. *Empirical Software Engineering*, *13*(3), 303–337. https://doi.org/10.1007/s10664-008-9065-9