CS 250 Final Project

Sprint Review and Sprint Retrospective

Role: Scrum Master

Christopher Meglen

Christopher.Meglen@snhu.edu

October 14, 2025

This paper presents the Sprint Review and the Sprint Retrospective for a two week sprint in the Vision Quest Software case. As Scrum Master I explain how Agile practices were applied, summarize the increment, evaluate communication and tooling, and document what we will change next sprint. The intent is to show evidence of disciplined Agile delivery and to demonstrate how communication practices improved transparency across two locations.

Sprint Review

Sprint goal:

Deliver a vertical slice for Customer Profile so a user can view and update name, email, and notification preferences in a single workflow.

Completed work:

We completed 7 of 9 stories that were planned. The increment included a responsive profile page, a secure update endpoint, optimistic concurrency checks, audit logging, analytics events, and localization for the form labels. The increment was demonstrated to stakeholders during the review.

Not completed:

Two stories were not finished. A reporting export was larger than expected once we tested the data schema. An accessibility story did not pass acceptance due to keyboard focus issues.

Stakeholder feedback:

Stakeholders accepted the increment and asked for a confirmation banner after a successful save and a revision to the privacy copy. They approved the approach for the deferred reporting work.

Flow and quality measures:

Velocity 28 points. Throughput 7 stories. Average cycle time 3.2 days per story. Median pull request review time 6 hours. CI pass rate 92 percent. Two defects were found and fixed inside the sprint.

Agile principles applied:

We used small stories with clear acceptance criteria, a single ordered backlog, WIP limits on the board, and trunk based development with short lived branches. Daily standups focused on flow and blockers. The review was centered on a working product increment, not slideware. These practices align with Scrum Guide guidance to deliver a usable increment each sprint and to inspect and adapt based on empirical evidence (Schwaber & Sutherland, 2020).

Interview Summary: Communication Successes and Barriers

I asked a developer and a tester for a brief reflection on communication. The developer said that the shared online board and short standups kept both locations aligned and made it easy to swarm when a story stalled. The tester highlighted early involvement in refinement. When acceptance criteria were written in Given-When-Then format, test design started days earlier and feedback reached developers faster. Both teammates noted one barrier. Threaded chat sometimes hid important decisions. We agreed to capture decisions on the board and to update the Definition of Done to include a link to the decision note. These observations match research that transparency and disciplined communication improve coordination in distributed Scrum teams (Kostin et al., 2023).

Review and Retrospective: Communication

What worked: a single shared board as the information radiator, daily standups that focused on flow instead of status, and timeboxed planning, review, and retrospective. What did not work: long chat back and forth on technical decisions and occasional gaps in updating the board when work started after hours. Action items: make the board the source of truth by requiring updates before standup, add an explicit Blocked state, and log decisions as short notes linked to the story. This approach follows evidence that visible work and fast feedback loops increase team performance and reduce burnout risk (DORA, 2024).

Review and Retrospective: Project-Management Tools

We used a lightweight tracker configured with one backlog and a sprint board. Columns were To Do, In Progress, Code Review, Testing, and Done. The tool was valuable because it reduced ambiguity and let both locations see the same truth. For the next sprint we will enable simple work in progress limits, require peer review before merge, and wire a continuous integration gate to run unit tests on every pull request. These steps support trunk based development and prevent long lived branches that hide integration problems. The change is consistent with findings that CI and small batches correlate with stronger delivery outcomes (DORA, 2024).

Review and Retrospective: Purpose of Agile Practices

Our goal is predictable delivery of small, high quality increments that provide value and learning. Scrum events exist to create regular inspection and adaptation. The review validated the increment with stakeholders and produced follow up requests. The retrospective focused on improving the system of work rather than blaming individuals. We validated that small stories, pairing on risky items, and a visible board are moving us away from the illusion of progress that the case described. Next sprint we will protect focus by reducing planned work by one story to leave explicit capacity for unplanned fixes.

Presentation: Explaining Agile Roles

Product Owner orders the backlog and maximizes product value. Scrum Master coaches the team on Scrum, removes impediments, and protects timeboxes. Developers plan and deliver the increment and maintain quality. Testers are embedded as developers who design and automate tests aligned to acceptance criteria. Stakeholders provide feedback in review and help refine goals between sprints.

Presentation: Explaining the Agile Process

We plan in small batches. Work flows from a single ordered backlog into sprint planning where the team crafts a sprint goal and selects stories. Daily standups focus on flow and impediments. Code is integrated continuously and reviewed by peers. The sprint ends with a review of a working increment and a retrospective that decides one or two changes to our system of work.

Presentation: Describing the Waterfall Model

Traditional waterfall sequence is requirements, design, implementation, verification, and maintenance. It relies on big batches and phase gates. This can create long feedback delays and an illusion of progress when documents move but the product does not change. Waterfall can still work for stable, low uncertainty domains with well understood risks, but it struggled in the Vision Quest case.

Presentation: Waterfall to Agile

The transition for Vision Quest replaces big up front plans with a single ordered backlog and short sprints. We slice features into small stories, use acceptance criteria to make them testable, and keep integration risk low with CI and short lived branches. Leaders support the change by attending reviews, protecting focus, and judging progress by working increments rather than documents. This is an incremental change strategy that fits the case and matches the Scrum Guide emphasis on empiricism and continuous improvement (Schwaber & Sutherland, 2020).

Next Sprint Action Plan

1) Add a Blocked column and make board updates a pre-standup habit. 2) Enable WIP limits and enforce peer review before merge. 3) Write acceptance criteria in Given-When-Then format during refinement. 4) Reserve capacity for unplanned work by planning one fewer story. 5) Schedule a cross site pairing window three afternoons a week.

References

DORA. (2024). Accelerate State of DevOps Report 2024. https://dora.dev/research/2024/dora-report/

Kostin, D., Al-Haddad, S., & Tell, P. (2023). Effective communication in globally distributed Scrum teams: A model and practical guidance. Australasian Journal of Information Systems. https://ajis.aaisnet.org/index.php/ajis/article/view/4501

Schwaber, K., & Sutherland, J. (2020). The Scrum Guide. https://scrumguides.org/

Limitations and Future Improvements

This sprint was a classroom scenario and not a production deployment. Metrics such as velocity and review time were included for realism, but they are illustrative. In a live project I would capture these from the actual tracker and CI system and compare trends across at least three sprints. For future improvement we will automate more tests on the profile workflow and add a simple deployment checklist so the definition of done reflects run time needs.

Conclusion

As Scrum Master I focused on transparency, short feedback loops, and small batches. The sprint produced a working increment, gathered feedback, and gave us concrete changes to our system of work. With the new working agreements and a clear next step, the team is positioned to deliver predictably while continuing to learn each sprint.

Applying Roles

Product Owner ordered the backlog and kept scope focused on a single slice. Developers designed and built the UI and service and maintained quality. The embedded tester designed acceptance checks during refinement. As Scrum Master I coached on small batches, protected timeboxes, and removed impediments. This role mix prevented single owner bottlenecks and spread knowledge across the two locations.

Completing User Stories

We used refinement to split large items, add acceptance criteria, and check dependencies. In planning we selected stories and a sprint goal. During the sprint we kept work visible on the board, limited WIP, paired on risky code, and merged daily to trunk with peer review and CI. Each story met the definition of done: review complete, tests passing, and demoable change. This flow moved stories from idea to working software.

Handling Interruptions

Mid sprint an external team changed the reporting schema. We paused the reporting story and created a spike to inspect the change. We protected the sprint goal by not pulling new scope and finished the profile slice. The spike produced concrete follow up tasks that we scheduled for the next sprint. This showed how timeboxes and a clear goal help a team continue delivering when direction changes.

Communication

Sample 1: short daily standups anchored to the board kept both locations aligned on flow and blockers. Sample 2: refinement notes captured Given When Then criteria so testers could start cases early. Sample 3: a decision note on the board summarized the schema change and the agreed path forward. These examples worked because they were specific, visible to everyone, and connected directly to the work.

Organizational Tools

We used one online tracker with a single backlog and a sprint board. Columns were To Do, In Progress, Code Review, Testing, Done, and Blocked. The tool was effective because it supported Scrum events. Planning pulled from the ordered backlog. Standups referenced the board. The review linked accepted stories to the demo, and the retrospective used board data for insights. For improvement we enabled WIP limits and a CI gate on pull requests to keep quality high.

Evaluating Agile Process

Pros in the SNHU Travel project: early feedback on visible work, the ability to change scope safely, and faster learning with small stories. Cons: discipline required to keep items small and to update the board, and the need for close product involvement. Based on the course experience, a Scrum Agile approach was the better fit for SNHU Travel because requirements evolved and fast feedback mattered. A strict waterfall plan would have delayed learning and increased rework risk.