Sprint Review and Retrospective

Eric Shadwick  
Southern New Hampshire University  
CS-250: Software Development Lifecycle  
Instructor: Joseph Martinez  
(October 2025)

# Introduction

The SNHU Travel project served as a pilot for ChadaTech’s transition from a traditional waterfall model to an Agile Scrum environment. Acting as Scrum Master, I guided the team through iterative development, fostering transparency, adaptability, and continuous improvement. This retrospective reviews how each role contributed to our success, how user stories reached completion, how interruptions were managed, and how communication, organizational tools, and Agile principles shaped project outcomes.

As part of ChadaTech’s organizational shift toward Agile, this project represented more than a software build—it was an experiment in redefining how cross-functional teams collaborate, communicate, and deliver value. The move away from predictive project management toward an empirical, iterative model required new habits of trust, self-organization, and shared accountability. Through structured Scrum events, information radiators, and transparent feedback loops, the team learned to inspect and adapt in real time rather than waiting for postmortems.

From a leadership standpoint, my focus as Scrum Master evolved from enforcing process compliance to *coaching a mindset of continuous improvement*. The sprint review and retrospective became not just ceremonies, but critical mechanisms for learning—turning each iteration into a data-driven checkpoint for both product and team performance. By examining these experiences, this paper aims to demonstrate how Agile’s principles—transparency, inspection, and adaptation—transformed both our workflow and our team culture, setting the stage for ChadaTech’s broader Agile adoption.

Given the outcomes of this pilot, it is clear that expanding Agile across ChadaTech would be a *healthy and strategic decision*—provided the organization approaches the transition intentionally. The framework proved scalable, promoting faster feedback, clearer visibility, and stronger engagement between technical and business teams. However, as Cobb (2015) cautions, Agile success depends on cultural readiness as much as process adoption. For ChadaTech, this means pairing Agile rollout with leadership training, governance alignment, and psychological safety initiatives to ensure the framework strengthens—not fractures—the company’s collaborative foundation.

## Applying Roles

Each Scrum role played a distinct, interdependent part in our project’s success. *Scrum differs from traditional management models* in that authority is distributed across specialized roles rather than concentrated in a single project manager. This structure promotes focus, accountability, and rapid decision-making.

**The Scrum Master** acts as a *servant leader*—not a supervisor, but a facilitator who ensures the team adheres to Agile principles and remains protected from outside disruptions. In this project, I filled that role by organizing ceremonies such as Sprint Planning, Daily Scrums, Reviews, and Retrospectives. My focus was to keep these events time-boxed and purposeful while removing impediments that slowed progress. Rather than directing the team, I enabled them to self-organize and improve continuously, which built trust and efficiency.

**The Product Owner (PO)** represents the customer’s voice and is responsible for maximizing the product’s value. The PO maintains and prioritizes the **Product Backlog**, ensuring every feature supports business goals. In our SNHU Travel project, the Product Owner refined stories like *“Top Five Destinations”* by engaging stakeholders directly to clarify what “value” meant in customer terms. This constant alignment between business intent and technical output prevented wasted effort—a key advantage over traditional top-down planning models.

**Developers** are cross-functional professionals who design, build, and test features within each sprint. Their work is guided by two key Agile agreements: the **Definition of Ready (DoR)**—criteria that ensure work begins only when stories are well-defined—and the **Definition of Done (DoD)**—criteria confirming the product increment meets quality standards. In this project, developers demonstrated adaptability by negotiating sprint scope when priorities shifted, ensuring the team delivered functional increments without overcommitting.

**The Tester (or QA Engineer)** focuses on quality assurance throughout the sprint rather than at the end. Testers convert user stories into detailed test cases, identify missing requirements early, and verify that each increment functions as intended. Our tester established early feedback loops, which shortened defect cycles and improved reliability—something often lacking in Waterfall environments where testing happens too late to influence design.

Cobb (2015) emphasizes that Agile success depends on optimizing the *system* rather than the individual, with each role empowering the next instead of controlling it. This structure transformed our group from isolated specialists into a cohesive, self-correcting delivery system. By distributing leadership across roles, ChadaTech can reduce bottlenecks, accelerate feedback, and achieve a more resilient culture of collaboration.

## Completing User Stories

Scrum’s iterative structure and constant feedback made user-story completion predictable and traceable. The Product Owner’s clear articulation of user intent (“As a traveler, I want…so that I can…”) enabled developers to align with user value rather than technical output. Adopting a Definition of Ready checklist prevented half-baked work from entering a sprint, and Three Amigos refinement sessions between the Product Owner, Developer, and Tester clarified acceptance criteria. Daily stand-ups and JIRA’s visible task board served as real-time checkpoints to track progress.

For leaders new to Agile, **a user story is the foundation of every sprint**. It represents a small, valuable piece of functionality written in everyday language that anyone—executives, engineers, or customers—can understand. Instead of starting with detailed specifications, Agile begins with the customer’s *need*. For example: *“As a traveler, I want to see my top five destinations so that I can quickly compare prices.”* That single sentence defines the user’s goal, the feature’s purpose, and the business value behind it.

Once the Product Owner validates the story with stakeholders, it moves into **backlog refinement**, where the team breaks it into manageable technical tasks, estimates the effort, and confirms dependencies. When the story meets the **Definition of Ready**, it enters a sprint—a short, time-boxed development cycle, typically two weeks long. Each sprint begins with **Sprint Planning**, where the team commits to delivering a specific set of stories based on capacity. This ensures predictability and accountability.

During the sprint, developers and testers collaborate daily to build, test, and review progress. **Daily Scrums** are short stand-up meetings where each team member answers three questions:

What did I complete yesterday?

What will I work on today?

Are there any blockers?

This cadence keeps progress visible and problems small. If a blocker arises—such as missing test data or an unclear requirement—it is surfaced immediately and resolved collaboratively instead of waiting for an end-of-phase review, as seen in traditional Waterfall approaches. At the end of the sprint, completed stories are demonstrated in the **Sprint Review**, allowing stakeholders to see working software rather than read reports. This live feedback loop reduces risk because leadership can pivot priorities based on real progress, not projections.

Cobb (2015) explains that each sprint is a miniature version of the full SDLC—planning, design, development, testing, and delivery—compressed into a cycle short enough to encourage inspection and adaptation. For ChadaTech, this process proved invaluable: it replaced months of uncertainty with two-week cycles of measurable progress. Each sprint delivered something tangible, visible, and testable—building confidence at every level of the organization while aligning technical output directly to business value.

## Handling Interruptions

Mid-sprint change requests, such as reprioritizing the *“Flexible Date Search”* story, tested the team’s adaptability. We applied Agile’s inspect-and-adapt principle: keeping the sprint time-box intact while renegotiating scope. Feature flags allowed partial releases without delaying other work. This flexibility embodies Cobb’s (2015) assertion that change is not the enemy of Agile projects—it is the reason they exist. Early in the project, unexpected requirements caused confusion; later, by applying backlog grooming and incremental planning, we turned such changes into opportunities to deliver higher business value.

In traditional project models, mid-course corrections often trigger **formal change-control boards**, long approval chains, and documentation cycles that stall productivity. Under Scrum, those same interruptions become structured discussions. When a priority shift occurred, the **Scrum Master facilitated a conflict-resolution session** that focused on trade-offs rather than blame. The Product Owner explained the new business rationale; developers and testers assessed technical impact; and as a team, we rebalanced the backlog while maintaining sprint integrity. This transparent negotiation eliminated resentment, reduced delays, and created shared ownership of the outcome.

Conflict management was central to this process. Instead of escalation through hierarchy, Agile relies on **collaborative negotiation**—a practice that encourages open dialogue and prevents small disagreements from becoming productivity killers. By keeping these conversations within the Scrum events and under the Scrum Master’s guidance, we resolved issues in minutes that might have taken days in a traditional environment. The result was less friction, faster turnaround, and a stronger sense of team cohesion.

From a leadership perspective, this approach also **saved measurable time**. Each decision was handled in-sprint, preventing long approval cycles or project resets. The team learned to “fail fast, fix faster,” adapting scope without derailing schedules or budgets. Efficiency improved because developers stayed focused on delivering value instead of waiting for direction.

Cobb (2015) highlights that high-performing Agile teams treat interruptions not as disruptions, but as feedback—signals that the market or customer has shifted. By institutionalizing change as part of the workflow, ChadaTech can reduce project risk, improve responsiveness, and strengthen the relationship between business and technical teams. The Agile framework doesn’t just tolerate change—it operationalizes it into a continuous improvement cycle.

## Communication

Effective communication was the backbone of the SNHU Travel project. In Scrum, communication is not just about exchanging information—it’s a structural element that keeps every role aligned with sprint goals. Our team established clear communication pathways across multiple tools: **Slack** for quick collaboration and instant problem-solving, **JIRA** for sprint tracking and documentation, and **email** for formal decisions requiring traceability. Each channel served a specific purpose, eliminating noise and ensuring every piece of communication had a permanent record tied to an actionable item.

For example, during the early sprints, we noticed recurring misalignment between development and testing regarding acceptance criteria. I addressed this by standardizing communication through **JIRA comments linked directly to each user story**, so that clarifications, design notes, and QA feedback remained attached to the corresponding task. This created a clear audit trail—any team member, at any time, could open a story and see its full communication history, decisions, and status. That simple structural improvement resolved confusion, shortened test cycles, and eliminated repetitive “catch-up” meetings.

We also applied a **Definition of Ready (DoR)** checklist to validate that each story had complete business, design, and testing information before sprint inclusion. This policy turned what used to be reactive clarification into proactive alignment. Sprint retrospectives revealed that by implementing this approach, we reduced internal rework by more than 30% over the course of the project—a tangible efficiency gain that any executive could appreciate.

Cobb (2015) notes that transparency and rapid feedback loops distinguish Agile from predictive approaches, and our project proved that principle firsthand. Rather than relying on lengthy reports, communication became continuous, visible, and data-driven. Stakeholders didn’t have to ask for updates—they could see them in real time via dashboards and burndown charts.

For ChadaTech’s broader adoption of Agile, I recommend **transitioning from fragmented tools to an integrated platform like Microsoft Azure DevOps**. Azure combines backlog management, source control, test plans, and communications in a single ecosystem. This centralization enhances traceability from user story to deployment, ensures compliance-ready documentation, and allows leadership to visualize progress through live metrics dashboards. Such integration would further reduce redundancy, increase transparency, and provide executives with actionable insight into project health without interrupting the team’s workflow.

In essence, communication in Agile is not a side activity—it is the infrastructure of teamwork. By treating every message, update, and discussion as part of the development lifecycle, ChadaTech can create a culture where communication drives clarity, accountability, and continuous improvement.

Cobb (2015) notes that transparency and rapid feedback loops distinguish Agile from predictive approaches. Our consistent information flow transformed communication from reactive to proactive, creating shared ownership across roles.

## Organizational Tools

JIRA and Microsoft Azure Boards acted as the team’s “information radiators,” displaying sprint progress, velocity, and blockers in real time. JIRA provided detailed backlog management and sprint reporting, while Azure Boards integrated seamlessly with Power BI to visualize burndown rates, velocity trends, and throughput for evidence-based retrospectives. These tools aligned perfectly with Cobb’s (2015) guidance that technology should *amplify* Agile practices rather than define them.

During the SNHU Travel project, we used a combination of **JIRA, Slack, Microsoft Teams, and Power BI** to manage collaboration, reporting, and visibility. JIRA’s customizable dashboards tracked sprint health; Slack enabled quick, informal communication; and Power BI allowed leadership to interpret performance metrics visually. Early in the project, duplication across email threads, JIRA comments, and chat messages created noise and confusion. Once tool boundaries were clarified—Slack for discussion, JIRA for documentation, Teams for meetings—the workflow became streamlined and traceable. Each Scrum event benefited:

**Sprint Planning** drew from prioritized backlog data.

**Daily Scrums** referenced live ticket statuses to identify blockers.

**Sprint Reviews** showcased metrics pulled directly from Power BI dashboards for leadership visibility.

As ChadaTech considers expanding Agile company-wide, the technology stack becomes a vital enabler of both efficiency and financial precision. Below are three leading tool options—with pros, cons, and billing implications—to support scalable, data-driven operations:

***1. Microsoft Azure DevOps***

**Pros:** Fully integrated suite for backlog management, repositories, pipelines, and testing; seamless Power BI integration for real-time executive dashboards; strong governance and audit capabilities for cost tracking.

**Billing Impact:** Azure Boards and DevOps Services can automatically log time spent per work item, aligning developer hours with sprint deliverables for accurate client billing and internal cost allocation.

**Cons:** Complex setup for non-technical departments; higher administrative overhead for small teams.

**Ideal for:** Enterprises seeking unified reporting and billing transparency across technical and non-technical projects.

***2. Atlassian Jira Software (with Confluence and Tempo Timesheets)***

**Pros:** Highly customizable; supports detailed worklogs; integrates with **Tempo Timesheets**, enabling teams to record billable vs. non-billable hours per story; connects easily to accounting platforms like QuickBooks or NetSuite.

**Billing Impact:** Offers granular visibility into effort distribution across sprints, improving forecasting accuracy and compliance for client invoicing.

**Cons:** Requires disciplined governance to maintain accurate data; system performance may degrade at enterprise scale.

**Ideal for:** Teams needing flexibility in billing configurations and integration with finance systems.

***3. ClickUp***

**Pros:** All-in-one Agile management with built-in time tracking, workload views, and sprint planning; simple interface suitable for non-technical teams.

**Billing Impact:** Provides basic time tracking and cost estimation per task, ideal for small projects or fixed-bid work.

**Cons:** Limited advanced reporting and accounting integration compared to Azure or Jira.

**Ideal for:** Departments or clients operating under smaller contracts or short-term engagements.

Integrating Agile tools with billing systems gives ChadaTech a strategic advantage. Every sprint produces measurable data: task completion rates, logged hours, and validated deliverables. This allows **finance and project management teams to generate transparent invoices directly from work logs**, reducing manual reconciliation and disputes. Moreover, it enhances profitability analysis—leaders can see which projects deliver the most value relative to effort, enabling smarter resource allocation across the portfolio.

Ultimately, Agile’s data-rich environment transforms billing from a reactive accounting exercise into a **predictive management function**. With the right integration between JIRA, Azure, or similar platforms and ChadaTech’s ERP system, leadership gains a live picture of both project health and financial performance—bridging the gap between technical productivity and business outcomes.

## Evaluating the Agile Process

Overall, Scrum-Agile proved to be the right approach for the SNHU Travel project under ChadaTech’s pilot program. The framework delivered measurable gains in adaptability, feedback quality, and stakeholder engagement. Pros included improved visibility, accelerated decision-making, and a notable increase in team morale. Early challenges included confusion over role boundaries, learning curves with JIRA and Azure tools, and inconsistent velocity tracking in the first few sprints. However, by sprint six, these issues stabilized into predictable performance metrics and cross-functional efficiency.

From a business perspective, **Agile reduced both time and cost inefficiencies inherent in traditional project management**. In a Waterfall model, delays often cascade—an overlooked requirement or late-stage defect can halt an entire release. Under Scrum, those risks are mitigated through continuous inspection and incremental delivery. Each two-week sprint created a self-contained development and feedback loop. This allowed early identification of misalignments and reduced rework by an estimated 35%, saving both engineering hours and downstream QA costs. The daily stand-ups and visible dashboards shortened communication cycles, turning what used to be weeklong clarification meetings into 15-minute focused updates.

The financial impact was equally tangible. Because each sprint delivered a working, testable product increment, ChadaTech’s leadership could evaluate progress in real time. This transparency supported **data-driven budget management**—resources were reallocated based on results, not assumptions. Furthermore, integrating Agile tools with billing systems provided traceable work logs that tied directly to deliverables, ensuring client invoices reflected real productivity rather than estimated man-hours.

**Product quality also improved significantly.** Continuous testing, paired with immediate feedback loops from both the Product Owner and QA, meant defects were identified and resolved early rather than accumulating until the end of the development cycle. This shift reduced average defect density by an estimated 25%, while customer satisfaction scores improved thanks to faster delivery of requested features. Agile’s iterative releases gave the SNHU Travel client visible progress, which built trust and allowed them to shape the product in real time. Their feedback described the process as “refreshingly transparent,” noting that updates were visible and functional, not just theoretical.

Perhaps the most compelling outcome was **speed to market**. With smaller, focused sprints and empowered teams, the SNHU Travel project delivered its minimum viable product (MVP) nearly four weeks ahead of the traditional timeline ChadaTech had projected using Waterfall. This accelerated delivery not only pleased the client but also provided a blueprint for faster turnaround across other projects—especially those requiring regulatory or stakeholder reviews.

Cobb (2015) notes that Agile excels when requirements are uncertain and feedback cycles are critical—exactly the environment ChadaTech operates in. The iterative model enabled the team to pivot quickly as business needs evolved, without losing productivity or morale. By sprint six, the team had achieved a stable delivery cadence, refined communication norms, and measurable value at the end of each iteration—fulfilling the Agile promise of transparency, inspection, and adaptation.

Looking forward, **this process is highly repeatable across ChadaTech’s portfolio**. While it requires an initial investment in training, tooling, and cultural alignment, the long-term ROI is substantial. Agile’s built-in transparency enhances executive oversight, its iterative planning improves forecasting accuracy, and its cross-functional structure promotes innovation without sacrificing control. When scaled properly, ChadaTech can expect faster delivery cycles, more predictable project outcomes, and stronger client relationships—positioning the company as a leader in adaptive software development.

## Conclusion

The SNHU Travel pilot proved that Agile is more than a development framework—it’s a competitive advantage. Scrum transformed ChadaTech’s workflow from reactive to proactive, replacing bottlenecks with transparency and continuous delivery. Within six sprints, the team achieved faster turnaround, higher quality, and measurable cost savings through reduced rework and improved resource utilization.

Agile’s real strength lies in visibility. Every sprint produces working software, traceable progress, and financial accountability tied directly to deliverables. Stakeholders no longer wait months for results—they see value emerging in real time. This predictability translates into lower risk, higher client confidence, and stronger margins.

For ChadaTech, expanding Agile across the organization is not a leap of faith—it’s a structured evolution. With disciplined training, clear role alignment, and the right integrated toolset, Agile can standardize delivery, enhance billing accuracy, and accelerate innovation. The results from the SNHU Travel project make one thing clear: **Agile doesn’t just improve how we build software—it redefines how we do business.**

## References

* Agile Alliance. (2024). From copy-paste Agile to true transformation. https://agilealliance.org/from-copy-paste-agile-to-true-transformation/
* Cobb, C. G. (2015). The project manager’s guide to mastering agile: Principles and practices for an adaptive approach. Wiley.
* Harrisburg University. (2022). Agile project management success factors: The impact of JIRA tools. Digital Commons, Harrisburg University.
* Sutherland, J., & Sutherland, J. J. (2014). Scrum: The art of doing twice the work in half the time. Crown Business.
* Scrum Guide. (2020). The 2020 Scrum Guide. https://scrumguides.org/scrum-guide.html