Simulating Industry: A Multi-Semester Agile Collaboration Study in an Upper-Level Software Development Course*

Augustus J. Scarlato
Computer and Information Sciences
Stetson University
DeLand, FL, USA
ascarlato@stetson.edu

Abstract

This paper presents a two-semester case study of an upper-level Software Development II course designed to simulate real-world Agile development within a teaching-focused institution. The course was unique in that it combined rotating Scrum roles, peer evaluations, and direct collaboration with a large not-for-profit healthcare organization over two separate semesters. Student teams worked on full-stack web and mobile app development while compiling structured sprints and presenting deliverables during three industry-led review sessions. Over the two semesters, 32 total students assumed various roles, including Scrum Master, Frontend Developer, Backend Developer, QA Lead, and Product Owner. During which time they received authentic client feedback aligning with real product requirements. The course design evolved between semesters as a result of industry feedback, peer evaluation trends, and overall course evaluation data. These results, which were reflection-driven, show improved technical quality, collaboration, and professional preparedness. The paper offers a unique and replicable model for integrating industry collaboration, Agile methodology, and intentional soft skill development into any software engineering curriculum.

1 Introduction

The demand for job-ready computer science graduates continues to rise, making colleges and universities face continued pressure to bridge the gap between theoretical coursework and real-world industry practice. Foundational concepts of computer science remain vital; however, today's graduates must possess hands-on experience with agile development cycles, team collaboration, and professional expectations that are demanded in modern software engineering. For professors of computer science, this creates both a challenge and an opportunity: how can faculty provide authentic industry-aligned learning experiences within the traditional undergraduate curriculum?

Our course, CS 321 Software Development II, was designed to address a portion of this challenge; however, this case study took it further. Officially described as a one-unit, project-based course, CS 321 aligns students with forming small development teams and building a full-stack product over the span of a single semester. Core topics include version control, requirements engineering, API development, UI/UX design, and clean coding practices. Students are required to demonstrate their work through presentations and collaboration milestones, following the completion of Software Development I.

This paper presents how CS 321 has been both modified and enhanced over two recent semesters to hyper-simulate a professional agile development environment. The instructional approach incorporates direct industry collaboration, providing students with structured exposure to a large not-for-profit healthcare organization during three key stages of their Project: initial idea refinement, mid-project feedback, and final project presentation. This partnership aligned students directly with the organization's information technology software development team. These industry touchpoints were designed to immerse students in real-world expectations, good, bad, or indifferent, with a primary focus on communication, scope, delivery, and iteration.

To support this learning model, the course uses a Monday/Wednesday/Friday meeting structure with weekly sprint cycles. Each in-class session serves as a daily standup, while groups also meet at least one hour onsite the scheduled class time, reinforcing team autonomy and accountability. Students rotate through core Scrum roles throughout this 16-week course: Scrum Master, QA Lead, Frontend developer, Backend developer, while managing tasks through tools such as GitHub, Jira, and Discord. The case study focused on two course sections, each capped at 20 students, where ideal teams align to five in order to balance responsibilities across the software development life cycle.

To that end, this paper investigates the impact of authentic industry collaboration, Agile role rotation, and continuous coaching on student readiness, performance, and perception. This case study outlines the course design, agile implementation, industry partnership model, and student outcomes across two

consecutive offerings of CS 321. It highlights key lessons learned, challenges faced, and repeatable strategies that other faculty can adopt to align their software engineering curriculum closer to industry practices, while also enhancing students' technical and professional readiness.

While project-based software courses are not new, this case study presents a rare blend of features: a multi-semester, fully Agile implementation embedded within a non-capstone undergraduate course; rotating Scrum roles that expose each student to leadership and technical decision-making; and a direct partnership with a real-world not-for-profit software development team. Unlike many other models that only simulate client collaboration, this course brought authentic product owners into the classroom, providing a high-touch, milestone-based feedback that shaped student deliverables in real time. To our knowledge, the combination of structure, realism, and replicability within a teaching-focused institution has rarely been documented at this level of depth.

2 Related Work

Software Development courses frequently have a primary objective to bridge the gap between academic theory and practical industry experience; however, the transition from classroom to workplace often is challenging for new graduates without the proper exposure [3]. Traditional software engineering courses often involve small, team-based projects where students follow a simplified development cycle, which entails requirements gathering to deployment [1, 8]. However, these projects lack the complexity and real-world constraints found in actual industry settings, which can lead students to be unprepared for the challenges of large-scale software development. Furthermore, the lack of real-world collaboration can hinder the development of crucial soft skills such as communication, project management, and teamwork, which are essential to success in the software development industry [7].

Agile and Scrum methodologies have gained significant attention in computer science education, specifically in project-based learning environments. The inclusion of software project management activities, when appropriately aligned with learning outcomes, can benefit students' progression and understanding of software engineering principles [4].

Full-stack development combined with teamwork and collaborative learning has also gained increased attention in recent computing pedagogy research. Kerslake and Karimi emphasize the benefits of practical, team-based projects for developing both technical and personal skills in full-stack coursework [5]. Chow et al. further argue that introducing rigorous test coverage requirements in full-stack web development can better align student experience with professional software practices [2].

While industry and academic collaboration remain relatively uncommon in Agile-focused teaching articles, a few documented capstone projects have bridged this gap. For example, in a Bachelor of Science in Information Technology (BSIT) program in the Philippines, Agile methodologies were integrated with real client partnerships. This integration enabled students to experience the dynamic nature of software development through close collaboration and a focus on iterative product delivery [6].

While such examples exist, most documented efforts describe single-semester implementations and often rely on mock clients or simulated project contexts. In contrast, this study presents a multi-semester classroom implementation featuring rotating Scrum roles, structured faculty mentorship, and authentic industry engagement. All of which advance the literature by offering a replicable mid-curriculum model grounded in professional practices.

3 Course Design

3.1 Grading Breakdown

Table 1: Previous and Case Study Course Grading Breakdown

Component	Previous Course	Case Study Course
Attendance & Participation (42 classes)	20%	20%
Project 1: Product / Presentation	40%	—
Project 2: Product / Presentation	40%	—
Code Reviews (16 weeks)	—	20%
Midterm Check-In	—	20%
Final Product / Presentation	—	40%
Total	100%	100%

4 Agile Methodology and Data Collection

This case study draws on two consecutive semesters of CS 321: Fall 2023 and Spring 2024. In Fall 2023, the course enrolled 19 students, organized into four teams (three teams of five and one team of four). In Spring 2024, 13 students were enrolled, forming three teams (two teams of four students and one team of five). This consistent structure enabled comparable application of the Agile framework across both offerings.

The continuation of the Agile methodology was implemented using the Scrum framework, with each week of the 16-week semester mapped to a full sprint. Monday sessions included sprint planning, backlog prioritization, task assignments, and short development during the remaining class time. Wednesdays were dedicated to active development for the entire class period, and Fridays were reserved for retrospectives, checkpoint reviews, and development. Each in-class meeting began with a brief standup led by the designated Scrum

Master, as it was a rotating role. In addition to structured class meetings, students were required to complete one hour of team collaboration outside of class per week.

Students rotated through all Agile roles for familiarity throughout the 16-week semester. Tools used to support these roles included GitHub (version control), Jira (ticket and backlog management), and Discord (team communication and collaboration).

The professor guided each team as a classroom-based Product Owner, providing continuous oversight and feedback throughout the class period. Weekly code reviews were also performed, utilizing GitHub's organization feature.

Each semester involved three structured meetings with an external industry partner: ideation, mid-project review, and final feedback. Qualitative feedback from industry partners was collected at the end of the course. Additional data sources included peer evaluations, GitHub logs, course evaluations, and instructor observations.

5 Results

5.1 Course Evaluations Results (completed by Students)

Table 2: Course Evaluation Questions and Results

Question	Historical	Fall 2023	Spring 2024
Overall Course Average (All Questions)	4.35	4.62	4.76
The class workload was rigorous.	4.16	4.35	4.82
The course material was presented in a clear manner.	4.24	4.71	4.82
The course was organized effectively.	4.24	4.71	4.82
The course challenged me to think deeply.	4.41	4.59	4.82
This course demanded intellectual effort.	4.56	4.65	4.82
The instructor was prepared for each class.	4.41	4.76	4.82
I am better able to communicate about the subject.	4.31	4.65	4.82
I learned a lot in this course.	4.38	4.65	4.73
The instructor advanced my knowledge.	4.39	4.47	4.73
The instructor explained how grades are determined.	4.38	4.47	4.64
Students' work was graded reasonably.	4.36	4.59	4.73
The instructor made helpful comments.	4.29	4.59	4.55
The instructor was accessible.	4.52	4.65	4.73
The instructor welcomed students seeking help.	4.57	4.71	4.82
The instructor is willing to meet with students.	4.52	4.76	4.73

Table 3: Professor Added Questions

Question	Fall 2023	Spring 2024
This course helped me understand software development in teams.	4.84	4.85
Working with an external client improved my motivation.	4.84	4.85
I feel more confident contributing to a professional team.	4.90	4.92

Table 4: Representative Student Feedback Quotes

Feedback	Theme
"Knowing that real software engineers would be reviewing our work made me take the project more seriously. It felt like more than just a grade, it felt like real job prep."	Industry Motivation
"Rotating roles helped me realize where I thrive and where I still need to grow. Being Scrum Master taught me how to lead a team, while working on backend gave me the technical depth I was missing."	Teamwork and Agile Methodology
"This was the first class that actually felt like working in a tech company. We had deadlines, meetings, code reviews, and client feedback. Everything was real and not from a textbook."	Real-world Readiness

5.2 Student Feedback Quotes

Impact Summary

- Increased student confidence in Agile workflows, sprint planning, retrospectives, and role rotation.
- High client satisfaction, with one Project adopted for beta release.
- Strong course evaluation scores, improving from ∼4.3 to 4.7–4.8+.
- Demonstrated technical proficiency: GitHub usage, API integration, RAG chatbot deployment.
- Growth in soft skills: leadership, communication, accountability.
- Scalable and cost-effective: GitHub, Jira (free), Discord, Figma.
- Clear replicability for other teaching-focused institutions.

6 Discussion

Bringing real-world industry partnerships into a project-based Agile course proved both beneficial and engaging. Feedback from students consistently highlighted the value of working with an actual client. Having a real-world audience raised the bar; from my observation, students also communicated more clearly, took ownership of their work, carried themselves with professionalism, and treated the experience more seriously than with a purely simulated scenario.

Several key changes were made between the two semesters based on opportunities during the first iteration. Role definitions were more clearly set and reinforced in the second semester, after some students in the first round blurred responsibilities and created confusion. The Software Development Life Cycle (SDLC), initially covered in a single lecture, was expanded to three sessions in the second semester to give students a firmer foundation. And after some teams took on more than they could deliver in the first run, more time was spent guiding students to set realistic goals and define minimum viable products (MVPs) from the outset.

6.1 Summary of Key Takeaways

Instructor Takeaways:

- A structured, high-touch mentorship model with GitHub organizations, code reviews, and peer evaluations supports transparency.
- Instructor involvement is substantial but yields measurable impact.

Student Impact:

- Students demonstrated growth in technical proficiency, communication, and leadership.
- Exposure to real product owners elevated their professionalism.

Future Improvements:

- Clarifying roles early.
- Expanding instruction on SDLC and MVP scoping.
- Simplifying tooling (e.g., transitioning to GitHub Projects).

7 Conclusion and Implications

Overall, this case study reveals that embedding real-world industry collaboration into Agile methodologies within a semester-long software development course is both feasible and impactful. This is especially true at teaching-focused institutions, where small class sizes are well-suited for rotating Scrum team roles, structured sprints, and continuous instructor feedback. These elements form an effective bridge between classroom theory and workforce expectations.

For colleges and universities with limited resources, this model offers a replicable framework, as success depends less on advanced infrastructure and more on intentional course design, clear expectations, and a willingness to engage with local or regional industry partners. Many organizations are eager to contribute to educational initiatives to give back, help shape the next generation, or gain fresh insights into the emerging workforce. Even a single virtual feedback session from industry professionals can have a remarkable impact on student focus, confidence, and engagement.

The structure of this study not only enhances students' technical ability but also reinforces communication, adaptability, and ownership. These traits, based on post-course feedback sessions, are increasingly valuable in today's software development workforce. As computing curricula continue to evolve alongside AI tools that automate much of the coding process, courses like this provide students with essential, real-world preparation that AI cannot replace. This model serves as a foundation for embedding experiential, practice-based learning into upper-level coursework.

In conclusion, the model outlined in this study supports both academic goals for students and faculty, as well as broader career readiness objectives. It encourages instructors to rethink their roles as facilitators of authentic experiences rather than traditional lecturers.

Most importantly, this case study offers a replicable model for embedding experiential learning into undergraduate curricula, one that prepares students not only to code, but to collaborate, communicate, and deliver results in industry.

References

[1] Andrew Begel and Beth Simon. "Struggles of new college graduates in their first software development job". In: *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*. 2008, pp. 226–230. DOI: 10.1145/1352135.1352218.

[2] S. P. Chow, T. Komarlu, and P. Conrad. "Teaching and testing with modern technology stacks in undergraduate software engineering courses". In: *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. 2021, pp. 241–247. DOI: 10.1145/3430665.3456352.

[3] Sandra Cleland. "Evaluating the Effectiveness of a Real-Life, Team Based Software Development Project for Tertiary Students". MA thesis. Curtin University, 2013.

[4] Javier González-Huerta et al. "Experiential learning approach for software engineering courses at the higher education level". In: *arXiv preprint arXiv:2012.14178* (2020). DOI: 10.48550/arXiv.2012.14178.

[5] Claire Kerslake and Omid B. Karimi. "Project-based learning of web systems architecture". In: *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. 2021, pp. 656–662. DOI: 10.1145/3456565.3460067.

[6] Gerard Ng and Ricardo V. Cruz. "Student and faculty adviser insights into an Agile methodology integrated into a Filipino company-sponsored IT capstone program". In: *International Journal of Hybrid Information Technology* 13.2 (2020), pp. 33–46. DOI: 10.21742/ijhit.2020.13.2.03.

[7] Alicia Radermacher, Gursimran Walia, and Debra Knudson. "Investigating the skill gap between graduating students and industry expectations". In: *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*. 2014, pp. 291–296. DOI: 10.1145/2591062.2591159.

[8] Thomas Way. “A company-based framework for a software engineering course”. In: *Journal of Computing Sciences in Colleges* 21.1 (2005), pp. 132–139. DOI: 10.1145/1047344.1047399.

Appendix A: Replication Summary for Faculty

- **Course Level:** Upper-division (Junior/Senior), ideally Software Engineering or Capstone
- **Class Size:** 10–20 students; optimal for 2–3 project teams
- **Project Length:** 16 weeks (1 semester); or two 8-week sprints
- **Team Size:** 4–6 students per team
- **Industry Involvement:** Minimum of 3 milestone meetings
- **Project Type:** Full-stack web or mobile app
- **Client Type:** Non-profit, startup, or university partner
- **Tools Required:** GitHub, Jira/Trello, Discord, Figma, React, Flask
- **Agile Framework:** Scrum-lite weekly sprints, rotating roles
- **Key Roles:** Scrum Master, QA Lead, Product Owner, Dev Team
- **Deliverables:** Sprint reports, retrospectives, MVP demo
- **Assessment:** Peer reviews, code reviews, client survey
- **Faculty Involvement:** Weekly mentoring, code review feedback
- **Prerequisites:** Programming, Git, intro web dev