A Near-Peer Mentorship Framework for Software Projects

Stan Kurkovsky
Central Connecticut State University
New Britain, CT, USA
kurkovsky@ccsu.edu

Chad A. Williams
Central Connecticut State University
New Britain, CT, USA
cwilliams@ccsu.edu

Abstract

The work describes a model for graduate-undergraduate mentorship that enhances the learning of both groups, while also enabling more opportunities for students to get real-world experience in a way that benefits the university's greater community.

ACM Reference Format:
Stan Kurkovsky and Chad A. Williams. 2025. A Near-Peer Mentorship Framework for Software Projects. In *Proceedings of the 30th ACM Conference on Innovation and Technology in Computer Science Education V. 2 (ITiCSE 2025), June 27-July 2, 2025, Nijmegen, Netherlands.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3724389.3731275

1 Introduction

One of the most pressing needs in the tech industry today is for graduates who not only possess advanced technical knowledge but can also effectively apply their skills in real-world environments. Our Computer Science (CS) program has successfully addressed this need by providing students with hands-on experiences through partnerships with industry and non-profit organizations. These collaborations have given our graduates a competitive edge in the job market. However, as enrollment in our CS programs continues to grow, so does the challenge of scaling real-world project opportunities to meet student demand. This challenge led us to develop a structured approach to expand experiential learning while maintaining its effectiveness.

Since 2014, our Software Engineering Studio has provided more than 600 students with hands-on software development experience, including collaborations with non-profits and community organizations. These service-learning opportunities, integrated into the capstone sequence of our BS CS program (Software Engineering and Senior Project), allow students to apply their skills in real-world contexts while strengthening essential professional competencies such as teamwork, time management, and communication.

Through this experience, we established a structured framework that facilitates engagement between student teams and project partners, maps project stages to specific student deliverables, and enables both formative and summative assessment of student work [2, 3]. This framework, designed for scalability and consistency across semesters, incorporates agile software engineering practices commonly used in industry. These include using product and sprint backlogs, sprint reviews and retrospectives, and velocity tracking to monitor project progress. Currently, this approach is used in two courses, involving approximately 50 students per semester.

2 Objectives

The near-peer mentorship [1] framework presented here addresses three key areas for improving our existing approach:

Bridging the transition to real-world projects: Undergraduate teams work directly with project partners, gaining firsthand experience with the challenges and uncertainties of large-scale real-world projects. While this experience is invaluable, the adjustment can be overwhelming. At the same time, research shows that peer or near-peer mentoring in professional contexts can significantly enhance student learning by complementing traditional coursework.

Providing technical leadership for non-profit projects: In corporate collaborations, experienced professionals often serve as Scrum Masters, guiding student teams by coordinating tasks, maintaining project timelines, and ensuring timely communication. However, non-profits typically lack the technical expertise to fill this role, as they rely on our students for their specialized skills. While these projects offer valuable learning experiences, undergraduate teams often struggle without structured technical leadership to coordinate their efforts and manage project risks effectively.

Developing technical leadership in MS students: Just as industry seeks graduates with real-world experience, it is equally important for students in our MS Software Engineering program to gain hands-on technical leadership experience. While leadership concepts can be taught in the classroom, MS students need opportunities to practice these skills with less-experienced developers. This helps them become more successful after graduation when many MS students will lead teams of junior developers, which requires them to have both technical leadership and the ability to mentor their team members in professional practices.

To address these challenges, we enhanced our software engineering project framework by integrating graduate students from our MS Software Engineering program as Scrum Masters for undergraduate teams. Many of these graduate students are already working in industry, making them well-suited for this role. Additionally, as many are alumni of our own undergraduate CS program, they are uniquely positioned to serve as near-peer mentors. While these MS students may have industry experience, they often enroll in our program to accelerate their technical leadership development. This near-peer mentorship framework benefits both groups: undergraduates receive the structured guidance needed for more complex projects, while graduate students gain hands-on leadership experience in a controlled yet realistic environment.

3 Methodology

In the Fall 2023 semester, we developed a comprehensive set of mentoring and professional guidelines for student scrum masters.

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/10.1145/3724389.3731275

The scrum guide [4] served as an excellent starting point. However, our guidelines took into account the very limited project management experience of our graduate students, the constraints and specifics imposed by the academic setup of the project (timing constraints, limited experience of student developers, meeting cadence, etc.), and the lack of technical background of many of our community-based project partners. We then selected a cohort of four current graduate students with the right background to serve as scrum masters. Preference was given to students with at least 6 months of industrial experience, and/or experience participating in a semester-long course-based project, ideally CS 410 or CS 498 from our undergraduate program. We asked each student to serve for two consecutive semesters in this role, for which they were rewarded with academic credits.

4 Near-Peer Mentorship Framework Features

Cohort continuity: Every semester we have been adding 3-4 new scrum masters to the cohort so that there is always a mix of senior and junior scrum masters in the cohort. This approach played a very significant role in knowledge transfer between the scrum masters.

The body of knowledge: After the end of the first semester, we asked scrum masters to anonymously contribute to a shared document organized as a collection of prompts that included advice for new scrum masters about supporting student teams with technical challenges, best practices for scheduling and running team meetings, encouraging team engagement and timely communication, and handling team conflicts and non-contributors. This document is being revised every semester to reflect the best practices of the framework.

Working with teams: Each scrum master met with their team on a weekly basis for a 30-minute "weekly scrum" which mimics a daily standup, but spans an entire week worth of the team's work. Student teams reported that the most important questions that the scrum masters helped them answer and understand included the definition and the specifics of the scrum master's role and the value that they can provide to the team, the best practices for writing user stories, as well as useful practices for encouraging project client interaction.

Continuous improvement: At the end of every semester, we conducted a "course retrospective" where the scrum master cohort made many observations and suggestions (many of them anonymous) to help improve the mentorship framework. This included actionable guidance for returning scrum masters (creating a welcoming atmosphere for the team by relating to their own experience, and having a solid understanding of scrum and agile practices), improving team meetings (using standardized technology for backlog management, discussing team velocity trends, and concluding with actionable takeaways), professional growth as a scrum master (building leadership and communication skills and reinforcing the importance of clear direction and team alignment), and proposed enhancements to the mentorship framework (e.g. providing basic git training and emphasizing the importance of all scrum ceremonies).

Starting in the Spring 2024 semester, we assigned each scrum master to one team of undergraduate students working on external projects embedded in CS 410 Software Engineering. All scrum masters met with the course instructor on a weekly basis, which helped gather regular feedback about what aspects of their work were successful and what areas needed improvement. These meetings also helped the cohort of scrum masters stay in sync with each other and with the course content. An evolving cohort of graduate scrum masters has been working with student teams ever since.

5 Student feedback

The feedback we received from students was overwhelmingly positive. While recognizing that scrum masters could not provide any technical help, students appreciated the support they received:

> Our scrum master kept us on track during the sprint as he required updates on what progress we made and our plan moving forward. If it seemed we were falling behind, he suggested how we may move forward. Our scrum master also assisted in sprint planning. Before each sprint began, we worked with our scrum master to decide which backlog tasks were most important for the sprint. He assisted in determining how to break down the backlog tasks into smaller items.

Graduate scrum masters were also pleased with the outcomes of this experience:

> Acting as a scrum master helped me improve some leadership and communication skills. Providing direction for a development team taught me how critical it is to convey clear and concise information for a successful project. Specifically, ensuring each team member was on the same page regarding their responsibilities while facilitating each meeting helped me improve communication with the team.

When asked, "What aspects do you like the most about being a student scrum master?" A graduate student responded "Exposing students to the same sort of Scrum-based team workflow that is all so common when working as a full-stack developer in the professional world."

6 Summary

This mentorship framework is unique in that it integrates graduate-undergraduate near-peer mentoring and software engineering leadership practice into the program curriculum, which provides tangible benefits to both graduate and undergraduate students, as well as external project partners.

Acknowledgments

This work was supported in part by NSF awards DUE-2029287 and DUE-2315322.

References

[1] Margery K Anderson, R Jerome Anderson, Laura S Tenenbaum, Emily D Kuehn, Holly KM Brown, Swati B Ramadorai, and Debra L Yourick. 2019. The benefits of a near-peer mentoring experience on STEM persistence in education and careers: A 2004-2015 study. *Journal of STEM Outreach* 2, 1 (2019), n1.

[2] SPSG Hub. 2025. *Scaffolded Projects for the Social Good.* https://spsg-hub.github.io/

[3] Stan Kurkovsky, Chad A. Williams, Mikey Goldweber, and Nathan Sommer. 2024. External Projects and Partners: Addressing Challenges and Minimizing Risks from the Outset. In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1* (Milan, Italy) *(ITiCSE 2024)*. Association for Computing Machinery, New York, NY, USA, 555–561. doi:10.1145/3649217.3653593

[4] Ken Schwaber and Jeff Sutherland. 2011. The scrum guide. *Scrum Alliance* 21, 1 (2011), 1–38.