WIP: Easing the Transition to Project-Based Software Engineering Courses with Externally Sourced Projects

Nathan Sommer
Department of Computer Science
Xavier University
Cincinnati, OH, USA
sommern1@xavier.edu

Stan Kurkovsky
Chad A. Williams
Department of Computer Science
Central Connecticut State University
New Britain, CT, USA
{kurkovsky, cwilliams}@ccsu.edu

Mikey Goldweber
Department of Computer Science
Denison University
Granville, OH, USA
goldweberm@denison.edu

2025 IEEE Frontiers in Education Conference (FIE) | 979-8-3315-0105-1/25/$31.00 ©2025 IEEE | DOI: 10.1109/FIE63693.2025.11328148

***Abstract*—This work-in-progress research-to-practice paper describes the process of adopting a framework for employing community-engaged service learning in project-based software engineering courses at a small private liberal arts school. Working on semester-long authentic projects for external clients is a valuable learning experience for students, but running a project-based course comes with numerous challenges for instructors. In this paper, we discuss the initial struggles involved in our efforts to develop a new project-based course and the iterative improvements made to our approach over four semesters. The adoption of a course framework that was initially developed at another institution greatly accelerated improvements to the course and has led to a collaboration to refine the framework to make it suitable for different institutional contexts. Our framework provides guidance in vetting projects, course structure, student deliverables, and project handoff.**

***Index Terms*—Project based learning, Service learning, Computer science**

I. Introduction

The Computer Science Department at Xavier University (XU) is a relatively small department with four faculty members and approximately 150 undergraduate majors, situated in a liberal arts setting. In 2021, when considering how to revise our curriculum to better serve students seeking careers in the software industry, we split our single software engineering course into two courses, Software Engineering I (SE1) and Software Engineering II (SE2). With this change, SE1 is able to cover more practices, tools, and techniques before students embark on a semester-long authentic software development experience for a real-world project partner in the project-based SE2 course.

Project-based learning benefits students by connecting theory to practice [1], developing professional skills through collaboration [2], and increasing student engagement [3]. For project-based learning to be effective, projects must be realistic and student-driven to a significant degree [4]. This presents instructors with the challenges of sourcing authentic projects that align with the course goals, and of balancing the amount of structure and guidance given to students. XU, a Jesuit Catholic institution, values community-engaged learning and service-learning, and our department tries to illustrate ways that computing can be used for the social good across our curriculum [5]. Thus, the natural choice for finding project clients in SE2 is to look to local non-profit organizations or community partners. Despite the advantages of community-engaged learning [6], [7], running a project-based course with external community partners brings additional challenges. Difficulties can arise in finding projects with appropriate scope that match student skill levels, in managing relationships with project partners that have their own time constraints and may not understand the technical demands of the project [8], [9], and in ensuring that students stay on track with development that adheres to the project requirements. In this paper, we describe the process of getting the SE2 course off the ground at XU, and how adopting a course framework initially developed at another institution and based on the software studio model [10] helped alleviate some of the challenges involved.

This work was supported in part by NSF awards DUE-2315322 and DUE-2315323.

II. First Offering of Software Engineering II

Prior to the first offering of SE2 in Fall 2022, we approached XU's Eigel Center for Community-engaged Learning to see if they had any leads on good community partners to serve as project clients. These discussions resulted in the idea of doing an internal project for the first iteration of the course, with the Center acting as the project partner.

At the time, XU had recently released a new strategic plan that called for more high-impact experiences for students, including community-engaged and service-learning courses. The idea for the course project was to create a system to track high-impact experiences available to XU students and to provide a means for students to discover experiences relevant to their interests. There were enough students in SE2 to form four teams of four or five students. Each team was responsible for a different component of the system: back-end, web front-

end, iOS front-end, and Android front-end. Students were assigned to teams based on their interests and prior experience.

The first iteration of the course faced several issues due to inadequate planning and course structure. The project partner was eager for students to have ownership and agency over the direction of the project since it was going to be students who would ultimately use the system. Because of this, we decided not to discuss too many requirements for the project ahead of the semester so that the students could be as involved as possible in the requirements gathering process; however, scheduling conflicts prevented the partner from attending class to discuss requirements until several weeks into the semester. This time was spent reviewing agile processes and investigating technologies, but in these initial weeks, the students did not have a clear idea of what was ahead of them, and there was not much enthusiasm.

For the initial requirements discussion, the students did not actively participate as much as we had hoped. The class was too large for many students to feel comfortable asking questions, and the result was mainly a conversation among the instructor, the project partner, and a small number of students. When there were questions from students, in most cases, the partner did not offer a strong opinion about desired features, but rather left the questions open for the students to answer on their own. During the discussion, the project partner mentioned that they could provide students with a list of high-impact experiences, but this list was not yet compiled, and it would be several additional weeks before the list was delivered.

Such vaguely defined requirements resulted in a chaotic development process. The back-end team was the least functional team in terms of teamwork, which was particularly problematic because they were tasked with defining the web API that would be used by the front-end teams. The front-end teams proceeded with assumptions about what data the API would provide once it was defined, but each team made slightly different assumptions and moved in different directions. By the end of the semester, there was still no communication between the backend and any of the front-ends.

Each team was required to use project management software to plan and track their progress, and during each class session, each team gave a brief update on their progress. Teams would demonstrate their progress to the class when there was sufficient progress to show, but the demos were spontaneous, and development was not strictly organized into sprints. Teams were expected to regularly plan their next steps, but the only manner of assessing this was through examining the state of their project management tool, and the teams were not required to submit planning documents at regular intervals. The teams that had a student emerge as a strong leader were relatively effective at planning their development without a strict structure, but the teams without a student to step up as a leader struggled to form effective plans.

Pedagogically, the course was not a complete disaster, and many of the students acknowledged that they learned a significant amount from the process. The front-end teams were able to design and implement user interfaces despite the lack of proper data from the backend. However, there was an underlying frustration throughout the semester, and the course was much less satisfying than it could have been.

The issues experienced during this offering of the course could be grouped into three broad themes:

- **Project planning:** Ineffective project planning, delayed partner involvement, and unclear requirements left students without direction early in the semester, resulting in a frustrating and chaotic development experience despite valuable learning opportunities.
- **Team collaboration and coordination:** Poor team coordination, lack of student engagement in early discussions, and weak leadership in some teams led to misaligned development efforts, backend dysfunction, and failed integration between project components.
- **Project management and course structure:** The absence of structured sprints, inconsistent use of project management tools, and lack of formal progress tracking resulted in unorganized development, difficulty assessing team planning, and ineffective feedback loops.

In retrospect, a number of better choices could have eased the frustrations of the initial offering of SE2. Thoroughly discussing the project scope with the partner ahead of the semester would have given the students a solid foundation to begin their work. Having example data compiled at the start of the semester would have made data modeling much easier. If each team worked independently on their own full-stack implementation of the system, the interdependence among teams would not have been an issue. Organizing the semester into fixed-length sprints with deliverables tied to each sprint would have resulted in more effective planning by each team.

III. Scaffolded Projects for the Social Good

Prior to the second offering of SE2, XU entered into a research partnership with Central Connecticut State University (CCSU). XU's role in the project is to adopt and help refine a course framework named *Scaffolded Projects for the Social Good* (SPSG) [11], which was initially developed at CCSU based on years of experience running project-based software engineering courses with community partners [12]. This partnership offered a chance to employ the framework in a different institutional context and refine it to be applicable at a variety of institutions. The framework provided precisely the structure needed to improve SE2 at XU.

The SPSG initiative provides structured resources, including guides, templates, grading rubrics, and case studies, to help faculty design and implement effective project-based service-learning courses. SPSG supports this goal by organizing projects into four distinct phases loosely based on the Unified Process [13]: Inception, Elaboration, Development, and Transition. These phases provide a structured yet flexible roadmap that guides students from initial problem definition to final project delivery. SPSG ensures that projects are well-scoped, effectively managed, and continuously refined through stakeholder collaboration. By following this approach, students gain hands-on experience in software development while engaging

with real-world challenges, fostering technical competency, teamwork, and a deeper understanding of computing's societal impact. The four phases help faculty scaffold learning experiences, ensuring that students receive the necessary guidance, resources, and opportunities for reflection at each stage.

Inception: The Inception phase lays the groundwork for the project before the semester begins. Faculty work closely with community partners to define the project's scope, goals, and expected outcomes. The primary deliverable in this phase is the **project proposal** document, which outlines the problem to be addressed, the project scope, relevant constraints, and final deliverables. This document aligns expectations among the faculty, students, and community partners, and creates a solid foundation for further project planning and development.

Elaboration: In the Elaboration phase, student teams begin collaborating with community partners to refine the project requirements and develop a detailed vision for the system's features. Key deliverables during this phase include the **requirements specification**, which documents both functional and non-functional requirements, and an initial prioritized **product backlog**, which outlines the features to be implemented in future sprints. Faculty guide students in developing these deliverables through collaboration with community partners, and the grading rubrics focus on properly refined user stories, estimation and prioritization, selecting stories for the first sprint, and sketching key user interface features. This phase sets clear expectations for the work to be done, ensuring all stakeholders share a common understanding of the project's objectives and deliverables.

Development: The Development phase follows an agile methodology, with student teams working in iterative sprints to develop and refine the system. Deliverables include **updated code repositories** and **sprint review and retrospective presentations**, as well as written **sprint reports**, which showcase completed features and any demonstrations of the system. Sprint reviews allow faculty and community partners to provide feedback and guidance to ensure that the system aligns with the partners' needs. The grading rubrics for this phase's deliverables emphasize reflection on lessons learned during each sprint and iterative updates to the project's requirements.

Transition: The Transition phase ensures that the completed system is successfully handed over to the community partner or to the next development team(s) and that knowledge transfer is effective. The primary deliverables in this phase include the **final product**, which is the fully developed, tested, and deployed software solution, and the accompanying **technical documentation**, such as installation guides, API references, and user manuals. The final **project presentation** involves students demonstrating the system to community partners, faculty, and potentially a broader audience, showcasing the final product, presenting technical documentation, and discussing the project's impact. This phase ensures the sustainability of the project and reinforces the connection between academic work and real-world outcomes. The grading rubrics for the technical documentation focus on detailed instructions for deploying the system and clear user-friendly explanations of the main features of the system's main features.

To support instructors interested in adopting SPSG, the authors have created a dedicated project website [11]. It includes a structured guide detailing the rationale, objectives, and prerequisite skills for each deliverable, reusable templates for students and instructors, and a comprehensive assessment rubric. Additionally, we provide insights on managing multi-semester projects across different institutional settings, and considerations for post-delivery project maintenance. The project feasibility section offers materials and rubrics to facilitate discussions with potential project partners and assess whether a project aligns with program curricula, timing constraints, student skill levels, and other relevant factors. Finally, we include a selection of completed student projects with the corresponding deliverables for each SPSG element.

IV. Second Offering of Software Engineering 2

For XU's second offering of SE2 in Fall 2023, we decided once again to partner with the Center for Community-engaged Learning for a new project. To avoid some of the issues of the previous year, we discussed more project details prior to the semester using the guidance of SPSG. During the **inception phase**, we decided that the students would build a web application through which the Center's community partners could post volunteer opportunities, enabling XU students to sign up for them. This would help the Center organize student volunteering, track engagement, and allow students to easily sign up to volunteer.

Using the SPSG **project proposal template**, we worked with the partner to create a project description, define the project scope, identify challenges, and specify deliverables. Intentional planning sessions also allowed us to think ahead about how the system would be hosted if and when it reached a mature enough state for deployment, and we met with XU IT staff to discuss options.

The Fall 2023 SE2 course was composed of 20 students who were divided into five teams of four. Rather than having different teams complete different portions of the project, which caused issues in the previous year, all five teams worked simultaneously on different implementations of the system, with the idea that the best implementation(s) would be used by one or more teams in subsequent semesters to continue the project. During the **elaboration phase**, the project partner came to class to discuss the different types of volunteer opportunities offered by various partner organizations and to answer clarifying questions. Each student team then defined their own set of requirements based on the discussion. As the teams organized themselves, they each created a **team agreement and information sheet**.

There were five two-week sprints during the **development phase**. Each week, the teams gave either a **weekly scrum update** to discuss their progress mid-sprint, or a **sprint review and retrospective presentation** with a demonstration, a discussion of what did and did not work well in the previous sprint, and the plans for the next sprint. Regular oral updates in class helped the teams share ideas and motivate each other, and

created opportunities to discuss ambiguous requirements when there were inconsistencies in the various implementations. Each team also delivered a **sprint report** at the end of each sprint, which allowed for more structured feedback on the students' planning process.

By the end of the semester, each student team produced a prototype that implemented a subset of the project requirements, yet no individual team produced a deployable system. The students were aware from the beginning of the semester that this was the likely outcome, and as the class approached the last sprint, it was emphasized that producing **technical documentation for the transition phase** was a high priority.

With the added structure of SPSG and the lessons learned from the first SE2 offering, the second offering was more satisfying for all parties involved and showed improvements in the following areas:

- Producing a **formal project proposal** based on iterative discussions during the inception phase ensured that important details were defined before the semester began.
- SPSG provided a **step-by-step structure** for the project, allowing students to follow a defined process for planning, executing, and refining their work, a significant improvement over the vague and undefined project planning in the previous year.
- The use of two-week sprints, regular scrum updates, and sprint retrospectives provided **consistent checkpoints** for tracking progress and adjusting work, offering much-needed structure compared to the lack of clarity in planning during the first semester.

Along with these improvements, there were some bumps in the process of adopting SPSG. The course structure was based on well-developed materials from CCSU, but some of the details about CCSU's approach were not explicitly defined. For example, the project proposal template is intended to be completed by the project partner initially so that project refinement begins purely from the partner's point of view. However, for this project, the instructor created the first draft of the proposal after initial discussions and sent it to the partner for refinement, leading to less direct input from the partner during the inception phase.

The semester also suffered from insufficient communication with the project partner in the elaboration and development phases. The elaboration phase deliverables were not immediately shared with the project partner, and understaffing at the community-engaged learning center's office made communication difficult. This combination caused a distance between the student teams and the project partner, leading to sporadic communication during the development phase.

The initial adoption of SPSG during the second offering of SE2 helped inform the creation of the comprehensive structure guide that explicitly defined aspects of the framework that were previously assumed, particularly in regard to planning and communication with the project partner, and the roles of the various student deliverables.

V. Project Continuation in Senior Capstone

XU's senior capstone course was already project-based prior to the creation of SE2, but students worked on projects individually or in pairs. Once it was clear that SE2 projects would often require multiple semesters of work, we decided to integrate SPSG projects into the capstone course as well. In Spring 2024, a team of 4 seniors continued the work that began in SE2 in the fall. During the **inception phase** for the spring semester, it was emphasized that the project partner needed to be more involved so that the students would receive more regular feedback.

Since the senior capstone team was given all five prototypes and associated documentation from the previous semester's teams, their **elaboration phase** looked quite different. They met with the project partner to discuss requirements, but this time there were existing prototypes to point to when discussing which features to keep, improve, add, or discard. The capstone students saw design decisions in all five SE2 implementations that they felt could be improved, so they began by re-designing the database largely from scratch and then incorporated elements from various SE2 implementations for the front-end. Overall, having reference implementations from which to build was extremely helpful for the students, and more frequent feedback from the project partner during the capstone course helped correct incorrect assumptions earlier.

By the end of this semester, we felt significantly more confident in our approach, achieving several important improvements:

- The project **partner's increased involvement** ensured more frequent feedback throughout the semester, allowing for quicker correction of misunderstandings and assumptions compared to the previous semester.
- Each SE2 team created comprehensive developer documentation to ensure a **smooth handoff** to the capstone team and maintain continuity between the semesters.
- The senior capstone team effectively used the SE2 reference implementations as a foundation for their redesign, enabling a more efficient development process and **building upon previous work**.

VI. Expanding the Number of Projects

For Fall 2024, we had enough students in SE2 that we needed two sections of the course, and so more projects were required. Having gained confidence and experience from working with an internal partner for the first three semesters, we felt prepared to expand to working with multiple external partners, and spent time in the summer of 2024 contacting local organizations to source projects. We also discovered more opportunities for internal projects, and ultimately we ended up running six projects for three internal and three external partners. While increasing the number of project partners added overhead in managing relationships, leveraging prior semester experiences led to an effective **inception phase**, with most of the overhead occurring before the semester began. There were at most 2 teams working on any given

project, which made communication with a given partner more straightforward, and in most cases the students handled communication with the partners themselves.

The **diversity of projects** was helpful for the students, both in terms of motivation and exposure to ideas. When every team was working on the same project in prior semesters, students did not feel they had as much ownership in the project, and there was more uncertainty as to whether their work would ultimately be deployed. Running multiple projects also allowed students to regularly see presentations from the other teams working in different domains, exposing them to a broader view of different types of software development.

VII. Conclusion and Future Work

Teaching project-based courses with external project partners involves many challenges, but offers an extremely rewarding experience for students. The challenges can be overwhelming for instructors embarking on this journey for the first time, and leaning on the experience of others can significantly ease the transition. The SPSG initiative packages effective practices based on many lessons learned into an **easily adoptable framework** that provides a solid yet flexible structure for various types of projects which may span multiple semesters in different courses.

Iteratively developing a project-based course with the use of SPSG significantly enhanced both the course structure and the quality of the projects. SPSG provided a **clear structure and framework** for the entire project lifecycle, which helped reduce ambiguity and confusion. Working within a well-defined framework led to better planning, clearer deliverables, and more manageable expectations for both students and project partners. SPSG **improved communication and collaboration** by emphasizing regular feedback from project partners and requiring proactive communication. This ensured that both students and partners were aligned on project goals, allowing for more effective problem-solving and quicker identification of issues. SPSG fostered **better design and development practices** by supporting an iterative approach, encouraging students to continually refine and improve their designs. Finally, SPSG ensured **smooth transitions between courses and teams** by emphasizing the importance of handoff documentation and well-defined deliverables. This allowed for continuity when the project moved from SE2 to the capstone course, making it easier for future teams to pick up where previous teams had left off. Overall, SPSG positively impacted XU's courses by improving planning, communication, collaboration, execution, and project continuity, resulting in a more effective course experience and higher-quality project outcomes.

Getting our project-based SE2 course off the ground was a challenging endeavor, but our established structure gives us confidence that we can continue to move forward. Most of our existing partners remain enthusiastic, and so students will continue to work on our ongoing projects. That said, each of our ongoing projects has an ambitious scope that has made it difficult to achieve full deployment, and so in the future we will seek projects that are more likely to be completed in one or two semesters of student work.

We will continue to refine SPSG and expand our recommendations as we gain further experience. Faculty at several additional institutions have expressed interest in adopting SPSG and joining the initiative, and their feedback and experience will improve SPSG's generalizability. We are also developing studies to measure the effectiveness of SPSG in developing students' technical and professional skills, as well as the effect on their attitudes towards computing for the social good. We have begun administering surveys to our students to produce quantitative longitudinal data in order to identify the most significant changes in students' dispositions after taking our courses, as well as qualitative responses to collect more targeted and comprehensive feedback than we find in course evaluations and student interactions. We have also begun collecting qualitative feedback from alumni who participated in courses using SPSG who are now working in the industry in order to better understand which components of the project-based courses were most impactful to them as they began their careers.

References

[1] B. Pérez and A. L. Rubio, "A project-based learning approach for enhancing learning skills and motivation in software engineering," in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, 2020, p. 309–315.

[2] D. Rosca, "Acquiring professional software engineering skills through studio-based learning," in *2018 17th International Conference on Information Technology Based Higher Education and Training (ITHET)*, 2018, pp. 1–6.

[3] P. Morais, M. J. Ferreira, and B. Veloso, "Improving student engagement with project-based learning: A case study in software engineering," *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, vol. 16, no. 1, pp. 21–28, 2021.

[4] J. W. Thomas, *A Review of Research on Project-based Learning*. The Autodesk Foundation, 2000.

[5] R. Connolly, "Why computing belongs within the social sciences," *Communications of the ACM*, vol. 63, no. 8, pp. 54–59, 2020.

[6] J. Davis and S. A. Rebelsky, "Developing soft and technical skills through multi-semester, remotely mentored, community-service projects," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 2019, pp. 29–35.

[7] N. N. Arony, K. Devathasan, Z. S. Li, and D. Damian, "Leveraging diversity in software engineering education through community engaged learning and a supportive network," in *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, 2023, pp. 247–258.

[8] M. Welch and S. Plaxton-Moore, *The Craft of Community-Engaged Teaching and Learning: A Guide for Faculty Development*. Campus Compact, 2019.

[9] S. Kurkovsky, "Student reflections on service-learning in software engineering and their experiences with non-technical clients," in *Proceedings of the ACM Conference on Global Computing Education*. New York, NY, USA: Association for Computing Machinery, 2023, p. 112–118.

[10] C. N. Bull, *Studios in software engineering education*. Lancaster University (United Kingdom), 2016.

[11] Scaffolded projects for the social good. [Online]. Available: https://spsg-hub.github.io/

[12] S. Kurkovsky, C. A. Williams, M. Goldweber, and N. Sommer, "External projects and partners: Addressing challenges and minimizing risks from the outset," in *Proceedings of the 2024 Innovation and Technology in Computer Science Education (ITiCSE 2024), July 8–10, 2024, Milan, Italy*. New York, NY, USA: Association for Computing Machinery, 2024.

[13] I. Jacobson, G. Booch, and J. Rumbaugh, "The unified process," *IEEE software*, vol. 16, no. 3, p. 96, 1999.