Scaffolded Projects for the Social Good: Broadening Participation in Service Learning for Computing Curricula

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

Michael Goldweber
mikeyg@denison.edu
Denison University
Granville, OH, USA

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

Nathan Sommer
sommern1@xavier.edu
Xavier University
Cincinnati, OH, USA

Abstract

Recent reports emphasize the need for a shift in undergraduate Computer Science (CS) education towards competency-based learning and the integration of ethical and responsible computing practices. Community-based service learning (CBSL) is a proven strategy to achieve these goals. However, despite a strong track record, service learning can be challenging to implement and may not always guarantee successful student experiences.

We introduce Scaffolded Projects for the Social Good (SPSG), a framework resulting from 10+ years of iterative refinement of running externally sponsored team-based projects in a software studio environment. SPSG aims to guide educators through the complexities of CBSL by addressing critical factors that have been consistently cited as barriers to the successful adoption of CBSL: project scoping, skill matching, managing project timelines that extend beyond a single term, community partner relationships, and project handoff and maintenance. This paper outlines SPSG's benefits over traditional methods and provides a sampling of student experiences in several recent SPSG projects. We reflect on how the framework can enhance student competencies through iterative formative feedback associated with a comprehensive set of project deliverables.

CCS Concepts

• **Social and professional topics** → **Computer science education**; **Software engineering education**.

Keywords

Software engineering studio; software engineering; experiential learning; service learning; course projects; capstone projects; external project partners

ACM Reference Format:
Stan Kurkovsky, Michael Goldweber, Chad A. Williams, and Nathan Sommer. 2025. Scaffolded Projects for the Social Good: Broadening Participation in Service Learning for Computing Curricula. In *Proceedings of the ACM Global Computing Education Conference 2025 Vol 1 (CompEd 2025), October 21–25, 2025, Gaborone, Botswana.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3736181.3747128

CompEd 2025, October 21–25, 2025, Gaborone, Botswana

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/10.1145/3736181.3747128

1 Introduction

Three recently published reports aim to refocus undergraduate CS education. Computing Curricula 2020 [11], shifts from a knowledge-based to a competency-based approach, emphasizing four components: knowledge (know-what), skills (know-how), dispositions (know-why), in the context of a given task–aligning with the needs of industry-bound graduates.

The second report, "Piecing Together the Next 15 Years of Computing Education" [9] notes that generative AI tools have the potential to dramatically impact not only *what* we teach, but *how* we teach and assess student success. It emphasizes computing's deep societal ties [6] and the responsibility of CS departments to prepare ethical professionals. The report advocates for integrating humanities and social sciences into CS curricula to broaden perspectives and foster collaboration with non-CS students.

The final report, CS2023 [18], highlights that the field has grown too vast to cover all aspects comprehensively within an undergraduate curriculum. It advocates the "less is more" approach [13] focusing on programming competency, building a correct model of how computation proceeds, and understanding (and practicing!) ethical and responsible behavior as computing professionals.

There are various strategies for incorporating the recommendations from these reports into a computing curriculum. Here, we focus on using community-based service learning (CBSL).

This paper introduces SPSG, a framework designed to guide educators through the complexities of CBSL. The primary goal of this paper is to demonstrate how SPSG addresses the critical challenges of implementing CBSL, such as project scoping, skill matching, managing extended project timelines, community partner engagement, and project handoff and maintenance. By providing a structured approach, SPSG enhances student competencies through iterative feedback and real-world project engagement. The paper discusses the phases of SPSG, its advantages over traditional approaches, and the positive impact on student learning outcomes, making it a valuable resource for educators seeking to integrate CBSL into their curricula. We also provide a comparison between SPSG and HFOSS [28] as they both aim to facilitate service learning in an authentic context.

2 Reasons to Focus on CBSL

Community-based learning is an emerging term that encompasses a wide range of learning activities where students and community partners collaborate in a mutually beneficial way. **Service learning** involves students applying their skills or expertise to projects that directly benefit a partner or community. In CS, service learning projects often result in deliverables such as custom-built software applications or contributions to open-source projects. **Community-based service learning (CBSL)** is a subset of service learning with a community partner, usually a non-profit, whose mission is to improve the social, environmental, or economic situation for community members. For CS students, this generally involves working in a team setting to develop software for a non-profit community partner. CBSL can address the recent curricular recommendations as outlined below.

The practice of software engineering (SE) clearly lends itself towards **competency-based computing education**. For many institutions, a SE course is the lone non-knowledge-based course in the curriculum, though capstone courses can also qualify. Furthermore, adopting CBSL assists in addressing the skills gap.

CBSL quite literally **takes students out of the classroom and puts them in their communities** where the impacts of computing, both positive and negative, are felt. The best way to learn to be an ethical, responsible computing professional is to practice being such a computing professional in an authentic setting.

One way to emphasize that **computing is inextricably linked with society** is to have students work on societal problems. In essence, this is the motivation behind the computing for social good in CS education (CSG-Ed) movement [14].

Developing a software application that conforms to stakeholder requirements often requires **additional learning about the problem domain**. This broadens students' perspectives and reinforces that applications in computing are interdisciplinary by nature.

While the full impact of **generative AI** on CS education is still unfolding, authenticity concerns have already pushed many courses back toward closed-book, high-stakes exams despite their well-documented drawbacks [12]. By contrast, CBSL largely avoids these issues.

Digital tools, including generative AI, boost productivity and connectivity while often reducing face-to-face interaction. CBSL counterbalances this by letting students practice professional AI use on real projects while **building relationship skills** through sustained engagement with community partners.

Finally, CSG-Ed in general and CBSL, in particular, has been shown to be effective in attracting students from groups traditionally underrepresented in the discipline [8, 15, 26, 27].

3 Challenges to Undertaking CBSL

Undertaking service learning in general, and CBSL in particular, presents significant challenges. In our outreach efforts, we have talked to numerous instructors who have tried and abandoned CBSL due to these challenges, or who would like to try CBSL but find the challenges too daunting. In no particular order, the primary barriers to more widespread adoption of CBSL are:

- **Identification of an appropriate non-profit community partner.** Potential partners lack the technical expertise or experience to be an *effective partner* in a software development project without additional assistance. This is especially poignant regarding partner time commitment, long-term data stewardship, and software hosting and maintenance. Additionally, if a partner's expectations are out of sync with the educational requirements of the course, it can impact the ability to achieve the desired student learning outcomes.
- **Identification of an appropriate project.** Partners lacking technical expertise may overestimate the scope of what a student team can accomplish over one or two semesters. It can also be difficult to find projects that align with the students' skills.
- **Time frame mismatch.** Interesting and doable projects rarely span just one academic term in length. This dissonance has led to numerous past incomplete CBSL projects.
- **Lack of institutional support.** Using CBSL calls for adopting a competency-based model in the relevant course. This may not be well supported by the adopter's department/institution. Furthermore, untenured/unprotected faculty may fear how both students and peers evaluate their non-traditional role in the classroom.
- **Time commitment.** Adopting any new pedagogical approach is time-consuming, as one needs to perform different, potentially unfamiliar tasks. With CBSL, time typically spent writing lectures, creating exams, and grading is traded for project team meetings, managing the community partner relationship, etc.

However, given that these unique CBSL opportunities are known to be highly beneficial to students (e.g. [3, 4, 7, 24, 29]), educators continue to pursue them in spite of their inherent challenges.

4 Scaffolded Projects for the Social Good

Scaffolded Projects for the Social Good (SPSG) is a joint effort by the authors to address the many issues related to incorporating CBSL into the computing curriculum and focusing computing education on student competencies [16]. During our academic careers, each author worked with many groups of students on software projects for external project partners. Collectively, by the end of 2024, we have collaborated with nearly 70 project partners who provided more than 90 distinct software projects to over 190 teams of upper-level undergraduate students.

Before collaborating on the SPSG project, each author independently adopted a studio-based approach [5, 25] to student software projects. This model supports larger, more realistic projects by extending beyond course limits and team constraints. In a software studio, teams work on multi-semester projects, enabling continuity, mentorship from industry professionals and peers, and hands-on learning. The approach simulates real-world software development, fostering collaboration, iterative improvement, role rotation, and continuous formative feedback.

SPSG emerged from a decade-long iterative refinement of a methodology designed to manage 10–15 externally sponsored, team-based software development projects each semester at a medium-sized regional university. Its goal is to formalize a software studio model to bring CBSL within reach of individual instructors and programs, regardless of the institution size or the level of support infrastructure that may be available at their disposal. In SPSG, each

project may span multiple semesters, with a different team contributing each semester. To that end, SPSG places a very strong emphasis on continuous formative feedback provided to the teams by the instructor and the project partner. As illustrated in Figure 1, this is achieved through a multitude of low-stakes deliverables spread throughout the semester that also facilitate knowledge transfer among teammates and across different teams.

The SPSG framework intentionally neither assumes student skill levels nor prescribes project scope, since these vary widely across institutions and courses. Used to adopt CBSL, SPSG gives first-time instructors a stable structure to calibrate scope without simultaneously redesigning the course. Because SPSG follows agile practices, teams regularly reassess goals; if a project proves too large for one semester, work can roll to a future team without disrupting expectations or assessment. By setting this norm at the outset, instructors define success as disciplined, iterative process rather than one-semester completion. Attempting CBSL without such a framework often forces mid-semester improvisation that leaves both students and instructors dissatisfied.

4.1 SPSG Structure

SPSG projects are grounded in agile principles, emphasizing short iterations, project partner engagement, and flexibility to adapt to changing requirements and challenges. Central to the SPSG framework is its structured approach to project management, visually represented in the SPSG Roadmap (Figure 1). This figure illustrates how each semester of a project is systematically organized into four distinct phases: inception, elaboration, development, and transition, each carefully tailored to the project's evolving scope. The SPSG Roadmap outlines these phases and details the specific activities, key deliverables, and their intended purpose and outcomes for each step, providing a comprehensive guide for both instructors and student teams.

Inception: This phase, conducted before the semester begins, involves the instructor collaborating with the project partner to assess feasibility, align the project with student capabilities, define scope and duration, and set expected outcomes. The result is a standardized project proposal for student teams. Since some projects may prove infeasible, ample time is allocated for iterative discussions with potential partners. The SPSG framework provides guided questions and a project feasibility assessment rubric to help instructors determine whether to proceed with a project.

Elaboration: Student teams work closely with the project partner to define requirements. For new projects, teams conduct multiple interviews to gather requirements, which are then translated into user stories for the initial product backlog. For ongoing projects, teams review and adjust existing user stories. Throughout this phase, teams collaborate with the project partner to ensure user stories accurately reflect desired functionality and are properly prioritized.

Development: This phase follows an agile methodology with two-week sprints. At the start of each sprint, teams select user stories from the product backlog. Given students' limited availability (up to 10 hours per week), daily scrum meetings are replaced with a weekly in-class scrum. Teams report progress, outline plans, and maintain regular communication with the project partner to address issues and clarify requirements. Each sprint concludes with a sprint review, where teams present their progress and receive feedback from the project partner. Additionally, in-class sprint retrospectives allow teams to reflect on achievements, discuss lessons learned, adjust the backlog, and refine their workflow.

Transition: During the final week of the semester, the focus shifts to knowledge transfer. Teams demonstrate the project to the project partner and the class. The project may continue with the same team, transition to a new team, or conclude with only maintenance remaining. To support continuity, teams prepare user and deployment documentation for both the project partner and future teams. Knowledge transfer is further reinforced through team staggering, comprehensive documentation, and near-peer mentoring from past team members.

SPSG provides a robust scaffolding of student deliverables throughout the semester, ensuring continued student engagement with ample opportunities for the instructor to evaluate student work and provide formative feedback. Regular scrums and in-class retrospectives allow for feedback between student teams, increase accountability, and add a heightened sense of community.

To assist instructors interested in adopting the SPSG framework, the authors created a dedicated project website [16]. This online resource serves as a practical companion to the SPSG Roadmap (Figure 1), offering a *structure guide* providing specifics about *every* deliverable shown in the figure: the rationale, objectives, possible prerequisite topics/knowledge/skills, a reusable template for student and/or instructor use, and a detailed assessment rubric. This direct support for each element in the SPSG Roadmap makes the framework highly actionable for educators. We also provide *instructor perspectives* on various options for tackling multi-semester projects in different institutional contexts and for post-delivery project maintenance concerns. The *project feasibility* section provides materials and rubrics for guiding a conversation with a potential project partner and evaluating whether a project might be a good fit, taking into account the program curriculum, timing constraints, student skill levels, and other aspects. We also provide a sampling of completed student projects along with the deliverables they produced for every SPSG element illustrated in Figure 1.

4.2 Comparison with HFOSS

SPSG shares similarities with Humanitarian Free and Open Source Software (HFOSS) projects [28]. While both approaches aim to facilitate service learning, SPSG distinguishes itself by having a significantly lower adoption threshold, addressing several shortcomings of HFOSS, and integrating both formative and summative assessments of student outcomes [10, 22, 31]. Unlike HFOSS, SPSG projects work directly with community organizations, non-profits, or similar entities, eliminating many other stakeholders typically involved in HFOSS projects (e.g. project maintainers) from the decision-making loop. Some key differences between SPSG and HFOSS are outlined below.

- **Human Interactions:** SPSG requires students to work directly with project partners, providing ample opportunities to learn about the potential impact of their software solutions on the community.

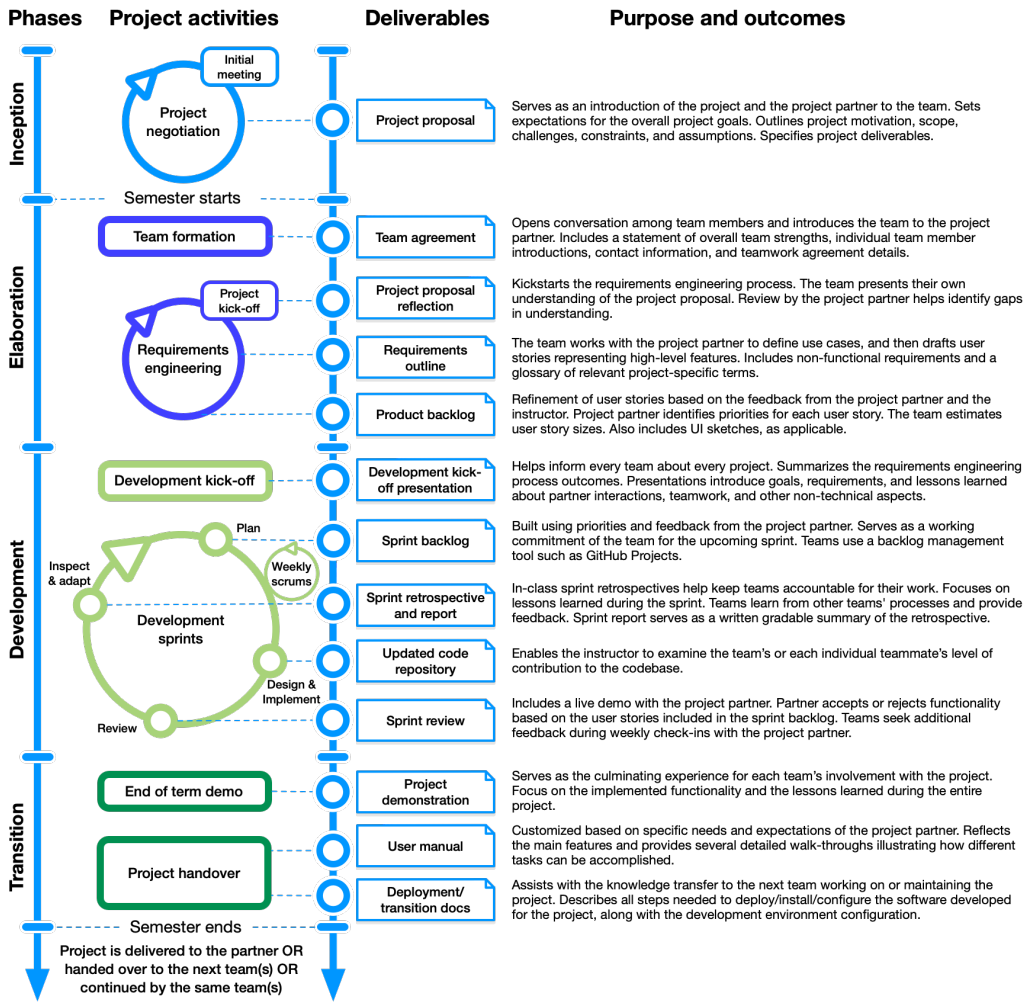

Figure 1: SPSG Roadmap: project phases, activities, deliverables, their purpose, and outcomes.

- **Non-Technical Skills:** SPSG values non-technical aspects of SE, such as communication, teamwork, and time management. Regular interactions with project partners help students improve their ability to communicate complex technical ideas to non-technical audiences.
- **Gentle Learning Curve:** SPSG's scaffolding incrementally guides students through requirements engineering, design, development, and testing. This synchronization with typical SE courses results in tangible student contributions that can be realistically assessed by the instructor, unlike HFOSS where students must simultaneously navigate learning the intricacies of the project combined with open-source project maintainer expectations that may diverge from instructor requirements and assessment needs.
- **Reduced Uncertainty:** SPSG helps manage scope by having the instructor iteratively negotiate project requirements with the project partner during inception. Consequently, project outcome assessment is fully within the instructor's control, without relying on external project maintainers to evaluate student contributions.
- **Timely Assessment:** SPSG helps align project work with course schedules while many HFOSS projects may depend on unpredictable response times from volunteer communities and maintainers. Requiring multiple deliverables provides instructors with ample timely opportunities to assess technical skills, teamwork, and communication beyond just accepted code contributions.
- **Project Completion Impact:** While the HFOSS model engages students to make incremental contributions to larger open-source projects that may be at various development stages, we believe that the SPSG framework supports a much higher rate of feature delivery, which frequently results in the project partners having some functionality available for testing within one semester. Many SPSG projects are completed within 2-3 semesters, delivering tangible value to the community partners.

5 Sample Application of SPSG

Currently, SPSG is being used by one of the authors to manage 10-15 student project teams per semester. Two other authors are

using it on a smaller scale with 1-5 projects every semester. This section illustrates how the SPSG framework was used with three recent projects undertaken by four teams of 4-5 students (A, B, C, and D) in the Senior Project course at a four-year medium-sized primarily nonresidential public institution [1]. All phases and deliverables of the SPSG framework, as comprehensively outlined in the SPSG Roadmap (Figure 1), were followed by the student teams and instructor.

Grey-box Project. Teams A and B contributed to an existing project for *Grey-box*, a non-profit organization based in Canada. Their solution, Project UNI, an Android device combining a WiFi access point with ample storage, enables offline access to free educational resources, e.g. Wikipedia and Khan Academy. It can be deployed in the most remote or imperiled communities. Previously, our teams implemented the functionality to download web-based content and preserve it for offline access. This semester, the teams focused on implementing role-based access control functionality for administrative and user-centric features of Project UNI and developing tools for analyzing and visualizing user activity logs.

Mon Ecole Project. Team C worked on a pilot project *Mon Ecole* led by our recent graduate from the Democratic Republic of Congo. In many countries, educational institutions still rely on paper-based systems to maintain student records, which are vulnerable to damage, loss, and deterioration. This could lead to inaccurate or incomplete data, limited accessibility, and restricted sharing of academic achievements beyond borders. This project aims to create a simple solution for instructors and administrators to digitize various paper-based artifacts using their smartphone cameras. These records would be cataloged and organized using local or cloud-based storage.

CT Explored Project. Team D collaborated with a nonprofit, *Connecticut Explored* [2], which has published a quarterly magazine on Connecticut history for over 22 years. They sought to expand their digital presence and interactive engagement through a mobile app focused on walking tours of Connecticut's historical sites. The app would integrate the magazine articles with walking instructions, maps, descriptions, and pictures from the articles, and allow users to post tagged images on social media to accumulate points on a leaderboard. The goal is to make information about the state's history freely available in an engaging format.

Comparative Analysis: The inception phase for each project spanned about two months. Initial conversations with the project partners (some of whom are non-technical) were followed by in-depth discussions using the SPSG methodology (described in detail in our prior work [21]) to assess project feasibility in terms of alignment with the technical capabilities of student teams, the academic calendar, and the partner's ability to work effectively with the teams. This helped us adjust the project scope and guide conversations with the partners. For the Grey-box project, we ensured the two teams worked on related but separate areas of functionality with minimal dependencies, reducing the risks of delays. With Mon Ecole, the goal was a minimum viable product (MVP) with stretch goals for additional meaningful work if the MVP features were easy to implement. With the CT Explored project, we focused on the core functionality of a walking tour as a cross-platform application and a single social media platform for the leaderboard. Once the project scope was narrowed down, the project partners wrote their proposals using a standardized SPSG template.

Student Engagement and Learning Outcomes: During the elaboration phase, upon receiving project proposals, teams worked closely with partners to outline requirements and reflect them in their product backlogs. They discussed the backlogs with the course instructor and presented them to other student teams. These reflections, along with feedback from the instructor and classmates, guided the refinement of their product backlogs.

The development phase was composed of six two-week sprints. The SPSG framework provided structural scaffolding, requiring teams to plan and implement a slice of the project functionality for each sprint while receiving continuous feedback. Unlike an HFOSS project, partners continuously reflected on the validity and relevance of each feature as it was being implemented and delivered, which was invaluable. Feedback from the instructor was used to resolve technical and architectural questions, while weekly scrums kept teams on track and encouraged honesty about setbacks. These interactions, along with sprint retrospectives, offered students ample opportunities to reflect on the lessons learned, increasing their SE competencies. One student noted:

> ...weekly check-ins [helped us] see and verbalize progress for others and gain feedback or hear/see different other people's questions about what is currently done from an outside perspective.

Challenges and Solutions: Frequent in-class presentations helped foster discussions about ethical considerations in real-world projects. Hearing questions from peers with different perspectives and learning about the challenges of different projects helped expose all students to a wider range of ethical issues, the complexity of decision-making, and the challenges of explaining technical ethics to non-technical stakeholders. By frequently presenting their ongoing work to classmates, teams had an opportunity to learn from peer questions, which may have encouraged them to think "outside the box" on design and implementation decisions and the importance of being able to explain their rationale. A student pointed out:

> It forces you and your team to prepare, discuss, and communicate on where you are in the project, what you are currently working on, any issues that are occurring, and where you will be going. Furthermore, other teams can ask questions and also learn from the good and the bad of other teams.

Social Impacts: Throughout the project, the SPSG framework helped expose students to many aspects concerning the societal impacts of computing, communication, teamwork, and ethics. We believe that by witnessing their project partners' dedication, the teams may have gained a deeper appreciation for **social responsibility**. All project partners were very passionate about the causes they and their projects served. Although this observation is purely anecdotal, we feel that this enthusiasm was contagious and that it imparted a lasting difference on the students.

The Grey-box and Mon Ecole projects focused on **developing computing solutions for populations in need**. During in-class discussions, many students pointed out that it was eye-opening for them to see how the technology we take for granted can significantly impact others' lives. Both projects addressed resource

Table 1: Student competency evaluation results

Team	Project	TD	CP	PR	Average
Team A	Grey-box	3.41	3.22	3.50	3.41
Team B	Grey-box	3.33	3.61	3.67	3.54
Team C	Mon Ecole	3.67	3.39	3.22	3.43
Team D	CT Explored	3.59	3.61	3.83	3.68

scarcity, including power supply, network connectivity, and digital infrastructure. All three projects addressed issues of **privacy and security** in maintaining and accessing computer records. With Mon Ecole, the "security by obscurity" of paper records was supplanted by digital records, which required careful consideration of access for authorized users. For the CT Explored project, the team had to take a deep dive into making an Instagram post on the user's device while ensuring that this post is properly tagged and captured by the application's leaderboard functionality running in the cloud while ensuring that no private information is disclosed.

Broader Impacts: Beyond the individual projects, the weekly/bi-weekly presentations can help create a ripple effect where exposing the class to issues across all projects can help broaden the impact of each project beyond its development team. This observation suggests a key benefit of this model: the **flexibility to choose a wider range of feasible community projects** when instructors can select projects where students will engage with different aspects of real-world issues and gain valuable lessons from the collective project pool. In essence, the model doesn't require every project to encompass every potential challenge. Learning opportunities are expanded as students encounter a wider range of issues through their peers' presentations. This flexibility may empower instructors to consider a wider range of community-based projects, enriching the overall learning experience.

Student Competency Evaluation: To independently evaluate student competencies, all four teams presented their projects at a semi-annual Senior Project Showcase. A panel of Industrial Advisory Board members and alumni evaluated student work on the projects using a standardized rubric to assess student attainment of learning outcomes aligned with professional competencies rated on a scale of 1 to 4 (unacceptable, marginal, proficient, exemplary), as summarized in Table 1. They included:

Technical Depth (TD): Is the solution well designed and incorporates all customer's needs and feedback? Does the project utilize a broad range of current technologies and tools? Do all team members demonstrate deep technical knowledge and skills that were applied in this project?

Communication and Presentation (CP): Was the presentation clearly organized with proper word choice, expression, and visuals? Was the team able to explain all aspects of the project and discuss the work in a broader context?

Professionalism (PR): Did all team members demonstrate adequate teamwork skills and substantively contribute to all aspects of the project? Did all team members exhibit confidence, respect, and an appropriate tone in all interactions?

It is important to point out that SPSG focuses mainly on team-based projects and deliverables. As such, all SPSG deliverables enable formative and summative evaluation and feedback at the project/team level. Since each SPSG project is hosted within a given course, we address individual assessment and feedback by utilizing relevant course-level assessment instruments: traditional quizzes/tests, self-reflection at the end of the semester, and/or peer review of individual contributions to the teamwork [23].

All teams delivered functional software to their project partners by the semester's end. With the Grey-box project, most of the functionality developed by teams A and B is currently being merged into the main project. Mon Ecole is evaluating the next steps for full-scale development beyond team C's pilot. Team D's software is undergoing usability testing with a small user group to identify issues and inform future feature development. Another student team is currently working on the CT Explored project with two previous team members serving as near-peer mentors.

6 Discussion and Future Work

We believe that SPSG represents a viable approach to address the challenges of transitioning to competency-based computing education by bringing real-world projects to the classroom and by helping expose students to the broader impacts of the computing profession on society at various levels of scale. Additionally, SPSG helps manage many projects conducted by multiple teams across several semesters. We believe that SPSG supports the "less is more" aspect of CS2023 [13, 18]: it helps foster student competencies, emphasizes practical problem-solving in an authentic context, and offers them an opportunity to practice being ethical and responsible computing professionals.

The SPSG framework is currently being used and refined at two institutions: a medium-sized public university (as discussed in the previous section) and a small private liberal arts college. We are planning to expand our community of practice and work with additional institutions to further refine the framework in a more diverse array of contexts. Feedback from schools with different student demographics, program specifics, and institutional contexts will allow us to adjust the framework and account for assumptions. The current framework, while refined through the authors' experiences at five institutions, has also been significantly shaped by insights from a broad group of instructors, including those shared during related Birds of a Feather sessions at SIGCSE 2024 [20] and SIGCSE 2025 [19], but continued refinement necessitates wider application.

Having exercised SPSG ourselves, we are confident that the framework's structure is helpful in navigating the instructional logistics of implementing CBSL. From anecdotal evidence through working with students, we believe that participation in CBSL projects has a positive impact on students beyond improving their technical and professional skills. Desiring stronger evidence of the latter, we are in the early stages of conducting a longitudinal study to measure the impact of student participation in CBSL projects on their formation and development of professional dispositions [17] and on their attitudes toward social responsibility [30].

Acknowledgments

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

References

[1] [n. d.]. Carnegie Classification of Institutions of Higher Education. https://carnegieclassifications.acenet.edu/

[2] Kathy Hermes (Ed.). 2024. *Connecticut Explored.* https://www.ctexplored.org/

[3] Nusaybah Abu-Mulaweh and William Oakes. 2018. Student learning in computing-based service-learning. In *2018 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–7.

[4] Alexander W Astin, Lori J Vogelgesang, Elaine K Ikeda, and Jennifer A Yee. 2000. How service learning affects students. *Higher Education* (2000).

[5] Christopher N. Bull and Jon Whittle. 2014. Supporting Reflective Practice in Software Engineering Education through a Studio-Based Approach. *IEEE Software* 31, 4 (2014), 44–50. https://doi.org/10.1109/MS.2014.52

[6] Randy Connolly. 2020. Why computing belongs within the social sciences. *Commun. ACM* 63, 8 (2020), 54–59.

[7] Randy W Connolly. 2012. Is there service in computing service learning?. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*. 337–342.

[8] Teresa Dahlberg, Tiffany Barnes, Kim Buch, and Karen Bean. 2010. Applying service learning to computer science: Attracting and engaging under-represented students. *Computer Science Education* 20, 3 (2010), 169–180.

[9] Adrienne Decker, Mark Allen Weiss, Brett A. Becker, John P. Dougherty, Stephen H. Edwards, Joanna Goode, Amy J. Ko, Monica M. McGill, Briana B. Morrison, Manuel Pérez-Quiñones, Yolanda A. Rankin, Monique Ross, Jan Vahrenhold, David Weintrop, and Aman Yadav. 2022. Piecing Together the Next 15 Years of Computing Education Research Workshop Report. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 2* (Providence, RI, USA) *(SIGCSE 2022)*. Association for Computing Machinery, New York, NY, USA, 1051–1052. https://doi.org/10.1145/3478432.3499037

[10] Heidi J.C. Ellis, Stoney Jackson, Darci Burdge, Lori Postner, Gregory W. Hislop, and Joanie Diggs. 2014. Learning within a professional environment: shared ownership of an HFOSS project. In *Proceedings of the 15th Annual Conference on Information Technology Education* (Atlanta, Georgia, USA) *(SIGITE '14)*. Association for Computing Machinery, New York, NY, USA, 95–100. https://doi.org/10.1145/2656450.2656468

[11] CC2020 Task Force. 2020. *Computing Curricula 2020: Paradigms for Global Computing Education.* Association for Computing Machinery, New York, NY, USA.

[12] Sarah French, Ashton Dickerson, and Raoul A Mulder. 2023. A review of the benefits and drawbacks of high-stakes final examinations in higher education. *Higher Education* (2023), 1–26.

[13] Mikey Goldweber. 2023. CS2023 in 3.5 Short Points. *ACM Inroads* 14, 4 (nov 2023), 44–50. https://doi.org/10.1145/3627170

[14] Mikey Goldweber, Lisa Kaczmarczyk, and Richard Blumenthal. 2019. Computing for the social good in education. *ACM Inroads* 10, 4 (nov 2019), 24–29. https://doi.org/10.1145/3368206

[15] Google. 2014. Women who choose computer science–what really matters: The critical role of encouragement and exposure. https://static.googleusercontent.com/media/edu.google.com/en//pdfs/women-who-choose-what-really.pdf

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

[17] Natalie Kiesler, Amruth N. Kumar, Bonnie K. MacKellar, Renée McCauley, Mihaela Sabin, and John Impagliazzo. 2024. Students' Perceptions of Behaviors Associated with Professional Dispositions in Computing Education. 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, 353–359. https://doi.org/10.1145/3649217.3653566

[18] Amruth N. Kumar, Rajendra K. Raj, Sherif G. Aly, Monica D. Anderson, Brett A. Becker, Richard L. Blumenthal, Eric Eaton, Susan L. Epstein, Michael Goldweber, Pankaj Jalote, Douglas Lea, Michael Oudshoorn, Marcelo Pias, Susan Reiser, Christian Servin, Rahul Simha, Titus Winters, and Qiao Xiang. 2024. *Computer Science Curricula 2023.* Association for Computing Machinery, New York, NY, USA.

[19] Stan Kurkovsky, Michael Goldweber, Chad Williams, and Nathan Sommer. 2025. Best Practices in Software Projects with Community Partners. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 2* (Pittsburgh, PA, USA) *(SIGCSETS 2025)*. Association for Computing Machinery, New York, NY, USA, 1722. https://doi.org/10.1145/3641555.3705099

[20] Stan Kurkovsky, Chad Williams, Mikey Goldweber, and Nathan Sommer. 2024. Community-based Service Learning: Best Practices in Software Projects with Community Partners. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2* (Portland, OR, USA) *(SIGCSE 2024)*. Association for Computing Machinery, New York, NY, USA, 1914. https://doi.org/10.1145/3626253.3635374

[21] 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. https://doi.org/10.1145/3649217.3653593

[22] Xiao Liang, Weiying Hou, Hongzheng Li, and Hongliang Liang. 2022. An Undergraduate Course for FOSS and with FOSS. In *Proceedings of the 2021 5th International Conference on Education and E-Learning* (Virtual Event, Japan) *(ICEEL '21)*. Association for Computing Machinery, New York, NY, USA, 161–167. https://doi.org/10.1145/3502434.3502465

[23] Joydeep Mitra and Eric Gerber. 2025. Examining Teamwork: Evaluating Individual Contributions in Collaborative Software Engineering Projects. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1* (Pittsburgh, PA, USA) *(SIGCSETS 2025)*. Association for Computing Machinery, New York, NY, USA, 777–783. https://doi.org/10.1145/3641554.3701821

[24] Barbara E Moely and Vincent Ilustre. 2014. The Impact of Service-Learning Course Characteristics on University Students' Learning Outcomes. *Michigan Journal of Community Service Learning* 21, 1 (2014), 5–16.

[25] Tom Nurkkala and Stefan Brandle. 2011. Software Studio: Teaching Professional Software Engineering. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (Dallas, TX, USA) *(SIGCSE '11)*. Association for Computing Machinery, New York, NY, USA, 153–158. https://doi.org/10.1145/1953163.1953209

[26] Michael Papadimitriou. 2014. High school students' perceptions of their internship experiences and the related impact on career choices and changes. *Online Journal for Workforce Education and Development* 7, 1 (2014), 8.

[27] Jamie Payton, Tiffany Barnes, Kim Buch, Audrey Rorrer, and Huifang Zuo. 2015. The effects of integrating service learning into computer science: An inter-institutional longitudinal study. *Computer Science Education* 25, 3 (2015), 311–324.

[28] Lori Postner, Darci Burdge, Stoney Jackson, Heidi Ellis, George Hislop, and Sean Goggins. 2015. Using Humanitarian Free and Open Source Software (HFOSS) to Introduce Computing for the Social Good. *SIGCAS Comput. Soc.* 45, 2 (July 2015), 35–35. https://doi.org/10.1145/2809957.2809967

[29] Mary Prentice and Gail Robinson. 2010. Improving student learning outcomes with service learning. (2010).

[30] Daniel S Schiff, Emma Logevall, Jason Borenstein, Wendy Newstetter, Colin Potts, and Ellen Zegura. 2021. Linking personal and professional social responsibility development to microethics and macroethics: Observations from early undergraduate education. *Journal of Engineering Education* 110, 1 (2021), 70–91.

[31] Therese Mary Smith, Robert McCartney, Swapna S. Gokhale, and Lisa C. Kaczmarczyk. 2014. Selecting Open Source Software Projects to Teach Software Engineering. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*. ACM, Atlanta Georgia USA, 397–402. https://doi.org/10.1145/2538862.2538932