Community-Engaged Software Projects: A Lightweight Approach

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

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

We describe a lightweight approach to community-engaged service learning, structured as a semester-long hackathon where students explore socially relevant problems through problem definition, ethical analysis, and rapid prototyping. By eliminating the logistical challenges of traditional service-learning projects, this approach provides a scalable way to integrate Computing for the Social Good into the curriculum while retaining key educational benefits.

CCS Concepts

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

Keywords

software engineering, capstone projects, ethics

ACM Reference Format:
Mikey Goldweber, Stan Kurkovsky, Chad A. Williams, and Nathan Sommer. 2025. Community-Engaged Software Projects: A Lightweight Approach. 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/10.1145/3724389.3731262

1 Introduction

The gold standard for the software engineering sequence and/or capstone course is community-engaged service learning where students, working in teams following an agile methodology, construct a software artifact that advances the mission of a not-for-profit organization. In the taxonomy of projects that fall under the label of Computing for the Social Good (CSG) [3, 4], such experiences are considered to be level 4, the highest level. However, these can be challenging undertakings since one must deal with thorny issues such as partner identification, project scoping, skills matching, project hosting, and long-term maintenance; to say nothing of non-traditional student assessment such an experience requires. We refer to such experiences as SPSG-projects after a time-tested framework, Scaffolded Projects for Social Good (SPSG) [5, 6], designed to aid adopters in overcoming the challenges these high-impact educational experiences present. This framework provides a highly-strictured approach to embed authentic service-learning projects into software development and similar courses, aligning them with course learning outcomes, and offering multiple avenues for formative and summative assessment of student work. In this techniques short paper, we describe a lightweight approach to achieving many of the outcomes from an SPSG-project that sidesteps the thorniest challenges such projects entail.

2 The Lightweight Approach

In a CSG level 4 project, students solve a real-world problem brought to them by an external (non-profit) stakeholder. The final deliverable addresses a problem with real-world benefits. The motivating idea behind this lightweight approach is to design a course to provide a CSG level 3 experience: solve a real-world problem as an exercise.

The basic idea is to run one's course as a one-term hackathon that has six phases. Similarly, this course has also been described as an experience in open-ended problem solving [7].

Theme Selection and Partner Identification: This phase happens before the term begins. The instructor selects a socially relevant broad topic of interest. The topic can be of local relevance or broader. Examples of this include: the opioid epidemic, food insecurity, aquifer depletion, homelessness, waterway pollution, etc. Based on the topic, the goal is to identify a local nonprofit working in the selected space willing to partner for the term.

This form of community engagement is one-directional. The community partner becomes a partner on the mission of educating students, but will not be receiving any software in return. As we detail below, given the light commitment on the part of the community partner, our experience has shown that most such partners are happy to participate.

This part is similar to an SPSG-project in that partner identification occurs prior to the term. However, there is no project scoping/negotiation, nor is there any need to manage partner expectations. Partners get to expose a new audience to their activities and to participate in students' growth as problem solvers learning how to apply technical skills to socially relevant issues.

Topic Education: Once the term begins, the first couple of weeks are spent doing a deep dive on the selected topic. In addition to reading articles, the class visits the community partner and/or the partner visits the class. This is part of the education phase; the role of the partner is to help educate the students on the various aspects of the chosen topic and what their role is in addressing the topic.

Topic education does not generally happen with SPSG-projects. With an SPSG-project students begin the term with an introduction to their community partner/client and an already agreed-upon project description.

Ideation: After students get up to speed on the problem domain, the next week or so is spent on project ideation. After an initial time to ideate, students are put into small groups. Each student pitches their idea to their group, getting feedback to refine their idea. In the next class session, students are put into different groups to repeat the process, which gives them an opportunity to refine their ideas with a different set of peers.

The ideation phase concludes with both a written report and an oral presentation. A key aspect is that in addition to a detailed description of their proposed software artifact, students must also present an ethical analysis of their approach. This includes a discussion of possible unforeseen consequences, potential negatively affected stakeholders, etc. There are several such frameworks one can adopt that guide students through this [1, 2, 9]. Finally, one might elect to hold the oral presentations in a forum open to both non-enrolled students, other faculty, and one's community partner(s).

Clearly, this phase has no counterpart with SPSG-projects where students are typically assigned to an already agreed upon project.

Project Selections and Team Creation: With software engineering project team size typically around six, the number of projects that will be selected is a function of the class size. As with hackathons, the *winning* projects are selected via a voting process. This can be restricted to class members only, or open to all who attend the oral presentation session, including the community partner(s).

Team formation comes after project selection. While instructors typically perform the team creation step in SPSG-projects, this lightweight version allows students to self-select onto teams using a preference voting scheme.

Software Development: This phase is the most similar to its SPSG-project counterpart. Students, working in small teams, using an agile methodology develop a software artifact. Unlike with a SPSG-project the final product is not production-ready software but more of a prototype or software that demonstrates a proof of concept.

The community partner's role is not that of a client, but a resource that student teams can periodically consult with - typically via prearranged online meetings. While the student – community partner relationship is not as formal as with an SPSG-project, it is nonetheless a worthwhile experience for students to interact face-to-face with people outside academia as part of a software development exercise.

Showcase: Typical of all such project-based courses is the public end-of-semester showcase event. Each team presents and demonstrates the artifact they created. Special attention is given to how each team addressed the ethical issues that were raised during the ideation phase.

Unlike real hackathons, there are no judges nor voting to select a winning project. However, after one showcase event, two teams received a small grant from a community partner to continue their project development over the subsequent summer.

3 Conclusion

This lightweight approach to community-engaged learning has some serious drawbacks when compared to SPSG-projects. First and foremost, SPSG-projects provide students the opportunity to hone their professional (soft) skills in an authentic setting. Face-to-face or real-time online interactions with an authentic project partner expecting an actual deliverable is a key learning outcome that our lightweight approach does not provide.

However, our lightweight approach has some significant advantages as well. Students involved in an SPSG-project spend all their time in the software development phase tackling technical issues. In a lightweight course, students not only see the social relevance of computing by working on a CSG project, they get to conceive of computing solutions to important problems. Concomitant to students practicing how computing can help solve important social problems, they also practice performing ethical analyses of proposed software solutions when it is most important; at the conception phase. For the most part, students are told what to program throughout their undergraduate career. There is great pedagogic value in students developing and publicly presenting their own ideas of what to create to address socially relevant issues [8].

While implementing a CSG level 4 SPSG-project offers many advantages, it may often require additional faculty effort and a certain degree of institutional support. Therefore, it is important not to underestimate the very real value of the lower bar required to undertake a CSG level 3 experience using a lightweight approach described here.

Acknowledgments

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

References

[1] Fatma Başak Aydemir and Fabiano Dalpiaz. 2018. A roadmap for ethics-aware software engineering. In *Proceedings of the International Workshop on Software Fairness* (Gothenburg, Sweden) *(FairWare '18)*. Association for Computing Machinery, New York, NY, USA, 15–21. doi:10.1145/3194770.3194778

[2] Markkula Center. 2025. *Markkukla Center for Applied Ethics.* https://www.scu.edu/ethics/

[3] Heidi J. C. Ellis, Gregory W. Hislop, Mikey Goldweber, Samuel Rebelsky, Janice Pearce, Patti Ordonez, Marcelo Pias, and Neil Gordon. 2024. Computing for Social Good in Education. *ACM Inroads* 15, 4 (Nov. 2024), 47–57. doi:10.1145/3699719

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

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

[6] 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

[7] Aletta Nylén, Mats Daniels, Ville Isomöttönen, and Roger McDermott. 2017. Open-ended projects opened up—aspects of openness. In *2017 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–7.

[8] Jacqueline Whalley, Michael Goldweber, and Harley Ogier. 2017. Student values and interests in capstone project selection. In *Proceedings of the Nineteenth Australasian Computing Education Conference* (Geelong, VIC, Australia) *(ACE '17)*. Association for Computing Machinery, New York, NY, USA, 90–94. doi:10.1145/3013499.3013508

[9] David Wright. 2011. A framework for the ethical impact assessment of information technology. *Ethics and information technology* 13 (2011), 199–226.