External Projects and Partners: Addressing Challenges and Minimizing Risks from the Outset

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

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

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

ABSTRACT

Software development projects sourced from external organizations can serve as an excellent platform to help build student competencies because they often provide an environment where students can practice applying their knowledge and skills in an authentic context. However, there are many challenges and risks that can jeopardize the successful execution of such projects. In this report, we discuss some of the lessons learned about the pain points encountered by computing faculty with over a decade of experience running a software engineering studio where teams of undergraduate students work on long-term projects sourced from external partners. Our experience is based on working with a mix of project partners with a major emphasis on non-profit and community organizations and non-technical project partners. We focus on a strategy to carefully screen prospective projects to reveal possible challenges in order to avoid or minimize risks that could impact student learning outcomes.

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, 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 Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2024), July 8–10, 2024, Milan, Italy.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3649217.3653593

ITiCSE 2024, July 8–10, 2024, Milan, Italy
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0600-4/24/07
https://doi.org/10.1145/3649217.3653593

1 INTRODUCTION

As outlined in CS2023 Computer Science Curricula [13], computing education is currently undergoing a transition from a knowledge-based to a competency-based approach. This transition requires computing programs to provide students with an opportunity to apply their knowledge and skills acquired in the classroom to solve practical problems in authentic contexts while demonstrating a range of professional dispositions. One way to provide students with such opportunities is by engaging them in externally sourced software engineering projects where they can solve real-world problems using current tools and technologies while interacting with real-world stakeholders and users who are unlikely to speak the same technical language.

We intentionally use the term 'partner' instead of client, customer, or sponsor when referring to a project's primary stakeholder because we view this relationship primarily as a partnership. While some programs approach externally sourced projects as sponsored work [17] for that entity, we believe that framing the relationship as a mutually beneficial partnership is key to ensuring that student learning outcomes receive sufficient priority. An ideal project partner, be it a small business or a nationwide company, an international non-profit or a local community organization, should be interested not only in the completion of the project, but also in working with students and helping them grow as professionals and as individuals. As computing faculty working with students, we align with the shared goals, which is why we are partnering with these external parties to help our students develop professional competencies, while helping the partner at the same time.

Over the years, the authors have accumulated a substantial amount of relevant experience working with external software project partners. Together, we worked with over 120 student teams who completed over 75 distinct projects sponsored by over 50 external partners. Most of these projects were offered in a studio format where overlapping sequences of teams work on projects spanning multiple semesters. This report documents various challenges of working with external partners we've experienced over the years, along with the solutions we found to address these challenges.

Here, we focus our attention on strategies to minimize the risks of experiencing these challenges from the very beginning of the project, ideally before any student teams are involved. Although it is impossible to avoid all challenges, we believe that many of them can be minimized through a careful project negotiation and screening process during which an instructor would try to learn more about the project partner and the scope of their project. As will become evident from the various challenges discussed below, it is crucial to emphasize when communicating with potential partners that the primary goals of the partnership should be focused on student learning and bringing mutual benefits to both parties.

This report is organized as follows. Section 2 outlines previous work in the field related to the topic, our software engineering studio model that provided context for much of our work, as well as some common experiences and pitfalls of working with external software projects. Section 3 describes common challenges that can be addressed by a careful screening process. Section 4 presents a list of screening questions that could serve as a backbone for interviewing a project partner. Section 5 concludes this report with a summary and our plans for future work.

2 PRIOR WORK

The authors accumulated a broad spectrum of experience working with various approaches to managing student software projects. While at Bemidji State University, a small public college in Minnesota, Williams led a year-long project where all student teams in Software Engineering I and II courses worked together on a single project for a local Boys & Girls Club. At Xavier University, a small private liberal arts college in Ohio, Goldweber devised an initiative [24] that enabled service-learning projects to persist across semesters ensuring the best possible match between the project needs, the capabilities of student teams, and the courses hosting these teams. Goldweber also led a 2-semester effort to develop mobile apps to assist with their city's opioid addiction crisis. Since 2014, Kurkovsky has been running a Software Engineering Studio at Central Connecticut State University, which is a large public school. The Studio connects external customers with teams of undergraduate students working on software development projects that involved over 100 teams consisting of 4-5 students in over 60 unique projects, many of which were done for non-profit or community organizations. Over the years, these projects helped form a highly scaffolded framework that provides enough structure to ensure project success while also being flexible, so teams had sufficient agility for adapting to evolving requirements or emerging challenges. At The College of Wooster, a small private liberal arts school, Sommer has advised over a dozen student teams as part of an 8-week summer program where teams of 2-4 students work for clients, both commercial and non-profit, and is now adopting the studio framework developed by Kurkovsky for use in Software Engineering II and senior capstone courses at Xavier University.

Each author followed a similar trajectory in implementing student software projects. Two decades ago, some of us started out by having one project completed by one team within one semester. To accommodate larger and more realistic projects and to enhance student learning, each of us independently gravitated to a studio-based approach where project size or duration is not constrained by the course length or the number of students. Many lessons were learned along the way. For example, some of our prior work [24] presents the results of an experiment to ameliorate the impedance mismatch over two different issues: student skill sets, and project length vs term length. The idea was for projects to float between both semesters and courses. A given project may be the focus of a database course in one semester and in a software engineering course the next. Not surprisingly, project hand-off in addition to a lack of universal faculty buy-in proved to be insurmountable.

A software studio [5, 19] can be characterized as an environment where a succession of student teams work on one or more projects that can span multiple semesters. It supports knowledge transfer from one team to the next, as well as mentoring of students by those with more experience, whether they are industry professionals, recent graduates, or students who are one year ahead. The software studio model works well to match the "6R" parameters stated by Bruegge et al [4] for successful software engineering projects, which include "a *real external client* who has a *real problem* to be solved with *real data*", where "students work together as a *real team* in a *real project* to solve the problem by a *real deadline*". We believe that a software studio model has a strong potential to help build student competencies [21] and help bridge the gap between computing education and the needs of the modern software industry [9].

Whether student projects are offered in a studio format or not, many current reports describing the challenges of external projects focus on technical problems experienced by the students (e.g. [6, 8]) or the instructors as they try to integrate external projects into their curriculum (e.g. [2, 6, 15]).

Working with external project partners is challenging for many reasons. Partners must understand academic constraints, especially concerning project cadence and timeframes [11]. The goals and expectations of being involved in the project are often misaligned: partners mostly want working software, while students mostly want good grades [12]. Many project partners do not understand what it takes to create working software of any appreciable complexity [7]. Some of these challenges can be minimized by involving alumni in helping source projects [11] and by emphasizing (or even enforcing) frequent interactions with external partners [15].

Engaging with external partners can occur in dedicated studio courses, software engineering, or capstone courses. In addition to the issues described above, reports on capstone courses also focus on structuring this experience as a quasi-co-op and/or platform to launch students in their professional careers [2, 18].

Steghöfer et al [23] describe a complex conceptual model for the analysis of external partners' involvement and participation in course projects and capstone experiences. As described, this model is suitable only for retrospective analysis and is not helpful to the instructors to make them aware of potential project challenges before the project commences.

HFOSS projects, although similar, present an entirely different collection of potential challenges. From the perspective of this report, the fundamental difference with HFOSS projects is that most of them lack a personified external project partner with whom the instructor would be able to discuss the project scope [3, 14, 20].

Many reports emphasize the importance of early discussions with a project partner regarding scope, cost, and schedule [19] because that's what most project partners usually want to know. While the conversation regarding the cost (free because all work will be done by student teams) and the schedule (aligned with semesters due to course alignment) is mostly trivial, the project scope deserves the most attention. As pointed out in prior work (e.g. [25]) student teams find it very difficult to plan and estimate the scope of a project. Compounded with the typical lack of understanding of the software development process by many project partners, project scope discussion is a key part of any negotiation process that needs to take place before starting any project. However, there are several other topics that we believe should be discussed with the prospective project partner to avoid or minimize various project challenges described below.

3 CHALLENGES AND IMPORTANCE OF SCREENING

In this section, we discuss various challenges of working with external partners, as well as solutions that worked for us to resolve these challenges. Each subsection begins with a "nightmare story" that we experienced at some point in the past and continues with a discussion of how such nightmare stories can be avoided.

3.1 Where Do Good Projects Come From?

A nightmare story: An instructor at a small liberal arts college committed to teaching a course involving team-based software development projects. They were told by the department chair that one or more projects would be provided in due time by the institutional office responsible for community outreach. However, as the start of the semester approached, it became clear that no such projects were forthcoming. How could this instructor make sure that they have a solid project before the class starts?

Some large institutions have the luxury of selecting from project proposals submitted by external partners that often do not have a presence in the local area (e.g. [16, 22]). Here, we focus on smaller institutions that may prefer to work with local partners. This is especially relevant in the context of service learning where students work with local nonprofits and community organizations to address needs that matter to their community or otherwise benefit the common good [10]. Such project partners may not have experience with project management or formulating their technical needs.

Identifying a good external project that matches the capabilities of the institution and the student teams is challenging in and of itself. However, before the project fit and the potential risks and challenges can be assessed, it is necessary to connect with a potential partner who might have a project. Our experience matches anecdotal evidence indicating that this may be especially difficult for the programs and/or instructors who have no prior experience with running such projects. We found that internal institutional projects were an excellent way to bootstrap our efforts that eventually grew into the software engineering studio infrastructure. Some published works (e.g. [11]) report a similar experience.

The authors have been successful in identifying external projects through various channels including respective institutional offices responsible for community engagement who are likely to have existing contacts with local non-profits, local government offices who can help get in touch with local community organizations, a local Society of St. Vincent de Paul who can help connect with charities, specialized websites focusing on non-profits e.g., greatnonprofits.org, colleagues who may be affiliated with various non-profits and community organizations or may lead research projects that include software development projects, respective industrial advisor boards and alumni who often bring business-focused project ideas with their employers, and personal connections.

3.2 How to Screen Prospective Projects?

A nightmare story (from a long time ago): Someone asked one of the authors, "How do you know when a potential project would be a good fit?" The author replied, "I just have a gut feeling about it." Is there a better way to evaluate a potential project that is based on a replicable process rather than on one's intuition?

Some institutions solicit external projects by asking prospective partners to submit a project proposal for consideration [1]. In our experience, many project partners find it difficult to explain the problem they are trying to solve and/or provide a reasonable amount of information to help us understand it. To bridge this gap, we ask all prospective project partners to have a conversation where they can explain their project ideas, and we can provide meaningful examples of completed projects relevant to the partner's operational domain and/or problems they are trying to solve. Such conversations almost always start with a real-time dialog, but then often transition to email, giving more time to provide additional information and answer follow-up questions that are very likely to emerge on either end.

The experience of having these conversations with dozens of project partners motivated us to create a comprehensive set of interview questions (discussed later in this report) that can help identify many of the possible challenges described here. Most importantly, these conversations help us work with the project partner to iteratively and collaboratively create a successful and realistic project proposal before the start of the semester. This provides a mutual commitment between the partner and the instructor and serves as a high-level project description given to the student team(s) before their first meeting with the project partner. It is not intended to serve as a detailed technical requirements document, but rather as an overview of the nature of the problem the partner's organization is trying to solve, what needs the project will address, and any other high-level information that would help the student team better prepare for the first meeting. It is also intended to serve as a springboard to help students formulate any follow-up questions they would like to ask their project partner as they start their requirements engineering process.

3.3 Is That a Realistic Project?

A nightmare story: A prospective partner working in the area of educational services for people with disabilities pitched a project idea to create a highly specialized system for learning and teaching support. They made an argument as to why none of the existing systems were a good fit for their needs. They also insisted on integrating many features into their desired product including a simple course management system, messaging, video conferencing, etc. When we tried to explain that many of these systems already exist and provide a free tier of service, the prospective partner continued to insist on their need for a dedicated software system precisely matching their needs. Our further arguments that this project would be very labor-intensive and may be beyond the capabilities of typical seniors were refused by the prospective partner who said, "But it all looks so simple on my iPad!"

In our experience, there is often a substantial disconnect between the prospective project partner's expectations and the capabilities of student teams. Many project partners do not understand the amount of effort and time that it takes to create working software, especially when it is developed not by professionals working full-time, but by college students with limited experience whom we expect to spend about 10-12 hours per week working on this project.

We believe that the best way to establish a good match between what is realistically doable and what the prospective partner is asking for is to provide a few examples of successfully completed projects that may be similar in scope or from a problem domain similar to that of the proposed project. This, however, presents a "Catch-22" situation for those instructors or programs without prior experience with external project partners. In our initial experience, we started with simple projects with reliable partners who were personal connections or were local to the institution, which helped minimize some of the challenges. We also made sure that each of the early projects had multiple tiers of goals including "must haves" and "could haves" (stretch goals), which ensured that at least some meaningful part of the project would be completed.

3.4 Does the Partner Know What They Want?

A nightmare story: One of the authors was working with a local Christian community center providing housing and recovery services. This prospective partner was interested in developing a solution to produce statistical reports about the people they serve and how well their needs were being met. We engaged in a conversation about what data is currently available, any existing relevant organizational processes that would generate this data, and how they can be automated. The partner indicated that there are no stable processes in place: there is no intake form, the intake process is an informal interview, captured data varies depending on which staff member conducts the interview, and all information is hand-written on loose paper. Furthermore, when asked what type of analysis they would want to perform (and, consequently, what data points they would need to capture on the intake form and elsewhere), the partner could not answer these questions.

The project partner must be knowledgeable and able to articulate their vision of the project scope in order for the project to be successful. We believe that employing active listening as the partner explains their vision can go a long way to help discover any potential ambiguities or omissions. Asking clarifying questions can help ensure that the prospective partner will be ready to provide appropriate guidance to the student team about the scope of the project and any relevant business/organizational processes. It is also very important to gauge whether the partner's vision is reasonably stable and it would not lead to significant scope creep. In our experience, the risk of scope creep is especially high with non-technical project partners who frequently assume that software development is easy and, if they change some of the requirements or add new ones, it would not jeopardize the project's success [15].

As with many real-world projects, the partner's vision is likely to evolve after the project starts, as additional details are learned about the required amount of work and/or new complexities are discovered. Having a core set of well-defined goals is still essential, even if the project vision beyond it remains vague.Sometimes, asking the right questions may lead to the necessity of declining a prospective project. On several occasions, when we asked a prospective project partner to describe the problem they were trying to solve, we were able to recommend an existing free or commercial software system that appeared to match their requirements reasonably well.

3.5 Are They Willing to Be a *Real Partner*?

A nightmare story: A non-profit was developing a mobile app to support community building among young adults. A substantial number of volunteer software developers managed by two volunteer scrum masters contribute to the project. Several of our student teams have been successfully contributing to this project for well over a year. Recently, a team reported that the scrum master missed a recurring meeting and stopped responding to email and Slack messages, which were essential to the team's progress. The non-profit leadership with whom this project has been negotiated was surprised by this lack of communication that lasted for three weeks, but they offered no help or a replacement scrum master. Eventually, the same scrum master returned with no explanation. As a result, an entire sprint's worth of work was essentially wasted due to a lack of coordination by the scrum master.

It is vital to ensure from the outset that the prospective partner sees this relationship as a *partnership* rather than a service that student teams will be performing for them. The partner needs to understand that they, or their staff, will need to be available to the student team for recurring weekly or bi-weekly meetings and keep an open line of communication for more urgent questions.

Most students will be unfamiliar with the partner's line of business, problem domain, or specific software. With or without the instructor's nudging, students will need regular input from the project partners, which will require a commitment of meaningful engagement throughout the project to answer student questions, resolve issues related to the project, and provide input on project-related decisions. To that end, the project partner must be willing to commit people necessary for the project's success. For example, someone with technical knowledge may need to be involved when the project is expected to integrate with existing technical architecture. Alternatively, on a project related to automating a business process, the partner should provide a description of that process and have the people most familiar with that process involved to answer questions that are very likely to emerge during the project.

3.6 Are They Ready to Go Back to School?

A nightmare story: In January, two of the authors were discussing the possibility of helping a local non-profit focusing on food insecurity. The partner was interested in developing a software solution to support Thanksgiving and Christmas food drives to mitigate a known issue of duplicate applicants to ensure that more families are getting food during the holidays. Our project pipeline was already full for the upcoming Spring semester, but we were happy to add it to the Fall semester pipeline, which, unfortunately, could not guarantee delivery of a reliably functional software solution in the timeframe leading up to Thanksgiving. The partner, however, assumed that we would have 10-11 months to work on this project and wasn't happy to hear about the realities of academic projects.

Regardless of the source of the project, it always takes a substantial amount of time and effort to ensure that the project is ready before it is shared with the students. Even if the project does not commence on the first week of the semester, we feel that it is very important to have complete clarity about every project offered to students before the semester starts. Whether our project partners like it or not, they have to understand the reality of the project being done by students enrolled in academic courses that are strictly aligned with academic semesters or terms. This rigid timeline requirement may not be suitable for some project partners.

Unlike full-time developers, each student can only be expected to spend 10-12 hours on the project every week. In addition to the typical semester timebox, there are other periods when little or no work will be done, such as any mid-semester breaks and mid-terms. While most students are aware of these "blackout periods" and usually account for them when they plan their workload, project partners often need to be reminded of these constraints. Also, unlike a real-world company, if a student drops the course leaving the team short-handed, it is not possible to "hire" new team members.

Partners need to understand that any delays on their end cannot be resolved simply by pushing the project delivery date into the future because this may have a negative impact on the students' ability to fully achieve their course learning outcomes. In terms of project delivery, the best-case scenario would imply producing partial functionality. In a more realistic scenario, delays might result in the project deadline extending to the end of the next semester during which another team could continue working on the project.

Different institutions may have different approaches to supporting multi-semester projects. Our software engineering studio framework accounts for projects that continue into the next semester with the same team(s) in a different course, or with one or more new team(s) in the same course. Regardless of whether the project continues or is completed, there needs to be clarity about who will be responsible for maintaining relevant project repositories and usage/installation/deployment documentation which we require our student teams to produce as a part of their project deliverables.

It is also very important to have a conversation with the project partner about any post-delivery maintenance requests and their timing. Unlike with commercially developed software, it is very difficult to provide timely bug fixes after the project is no longer in active development. One of the strategies includes placing any bug fixes or feature updates from one or several past projects into a standalone project in the project pipeline for the next semester. This is not always feasible because of a potentially long delay between the bug discovery and the date when a solution can be shipped. Another option that might work for an academic program with a high volume of projects is to have one team every semester "on stand-by" whose responsibilities would include addressing maintenance requests for past projects as they come in.

4 STRUCTURED SCREENING OF PROJECTS

In light of the myriad risks and challenges inherent in externally sourced projects, navigating the identification and assessment of these factors is key. Determining whether a project aligns with the desired student learning outcomes and, if so, implementing necessary mitigations underscores the need for a meticulous screening process taking place during one or more conversations, in real-time or asynchronously. Although every project and the respective project partner brings their own unique set of challenges, we found that a careful and methodical screening of every prospective project can avoid or minimize many of the challenges described above. To this end, we developed a set of interview questions that help us identify any red flags indicating potential challenges. These questions are a result of our project negotiation experience spanning about ten years and several dozens of partners with whom we worked on projects spanning between one and four semesters.

This section examines specific dimensions crucial to the project screening process, accompanied by targeted questions for potential partners, enabling an early feasibility evaluation. By systematically addressing these key aspects, our goal is to help instructors identify critical risk factors that may impact project success. This should empower instructors to anticipate potential challenges, devise effective risk mitigation strategies, and establish contingency plans.

4.1 Alignment with Course Goals

These questions are intended for the instructor to reflect on how the project aligns with the overall course goals and objectives, ensuring that it complements the curriculum and enhances students' learning experiences. Responses to these questions should aid the instructor in identifying any potential modifications that might be discussed with the partner for a more symbiotic relationship between the project and the educational objectives.

- How do you see this project aligning with the course goals and objectives?
- Can you identify specific aspects of the project that directly relate to learning experiences for the students?

4.2 Student Skill Relevance

The goal of these questions is to capture a sense of a partner's expectations of the students. Understanding the specific skills envisioned by project partners is crucial for customizing the project to align with both academic objectives and the technical needs of the partner. This understanding not only informs the instructors about the project's technical requirements but also provides insight into the partner's expectations regarding student capabilities. This contextual understanding ensures that the project aligns well with educational objectives and adequately prepares students for real-world scenarios.

- What specific skills do you envision students applying or acquiring through this project?
- Are there any skills that you consider critical for the project that might require additional learning time?

4.3 Workload Manageability

These questions are intended to get a sense of how compatible the project partner's timeline requirements are with the academic calendar and the sometimes unpredictable nature of student projects. These will help the instructor assess whether the expected timeline, end-product usage, and workload align effectively within the constraints of the academic setting. Understanding the partner's expectations regarding the final project's quality may expose a red flag if they do not align with the capabilities of the students and the academic goals of the course.

- What are your needs/expectations for the project in terms of timeline and expected use of the end product?
- Do you foresee any challenges if the project is behind schedule, and if so, what would be the impact?

4.4 Partner Engagement

The goal of these questions is to help uncover how likely the students will be able to get timely responses to their questions, a crucial aspect for effective student teams' progress. Partner engagement is pivotal in ensuring successful communication between student teams and project partners. These questions focus on evaluating the anticipated level of involvement from the project partner in providing guidance and clarification to student teams. Additionally, understanding if others will work with the student team and the extent of their involvement provides valuable insights into the overall support structure for student inquiries.

- How involved do you anticipate being in the project, particularly in terms of providing guidance and clarifications to student teams?
- Will there be anyone else working with the student team, and if so how do you envision their involvement?

4.5 Learning Clarity and Guidance

Some level of anticipated uncertainty and change can enhance students' understanding of non-academic project challenges. However, for a positive overall student experience, a clear vision is crucial for maintaining a stable learning environment. The clarity of the partner's overarching vision ensures that learning objectives remain on track, and consistent guidance is needed for student success. The following questions aim to explore the partner's perspective on how well-defined the project requirements are, their expectations regarding potential changes, and the identification of challenges that may be beyond their control.

- How well-defined are the project requirements from your perspective?
- What do you see as the aspect(s) that are most likely to change?
- What do you see as a potential challenge for the project that may not be fully within your control?

This question is intended to provide a sense of the project partner's familiarity with the normal challenges of software projects. A response indicating complete stability likely is not realistic, while an indication the project is in great flux could be a red flag.

- Are there aspects of the project that may undergo frequent changes, or do you expect a stable project scope?

4.6 Resource Requirements

These questions aim to identify any environment, resource, and long-term support needs, ensuring a clear mutual understanding that addresses these requirements to ensure project feasibility.

- Do you currently have an environment where the project should be developed or is expected to be deployed?
- How would you characterize the maintenance requirements for the project once it is implemented?

4.7 Privacy and Ethical Considerations

A vital, but often overlooked aspect of service-learning projects is their privacy and ethical considerations. This oversight is often exacerbated by the frequent lack of technical sophistication of non-profit partners. Issues that must be taken into consideration include:

- What type of personal data do you anticipate needing to be kept by the system? Is it appropriate for such data to be kept, and if so, what steps will be taken to protect it?
- Are there any privacy or sensitive issues associated with the project that you've identified?

4.8 Stakeholder Reliability

These questions are intended as a self-reflection for the instructor after the conversation(s) with a prospective project partner.

- How well does the stakeholder understand what they need?
- Can they clearly articulate the business problem that needs to be solved or the workflow that needs to be automated?
- How likely are they to be available to the student teams throughout the project?

5 DISCUSSION AND FUTURE WORK

The proposed approach to structured project screening has evolved over the years. It currently represents the experience of four instructors from three different institutions. Using the set of questions outlined above to screen potential projects would inevitably yield a unique set of answers specific to the project and the respective partner. These answers should always be considered in the local context specific to the instructor and their background, their course/program and institution, as well as their student population and their level of preparedness. We realize that each institution and academic program will likely encounter a set of unique challenges specific to their local context. However, we hope that the structured set of questions described above will be helpful as a starting point for any instructor working with a prospective project partner who is trying to uncover potential challenges and minimize their risks. In our experience, these questions always helped us highlight project aspects where we needed further clarifications or helped us identify meaningful strategies how the project scope could be adjusted to better align with the capabilities of student teams and their academic goals.

We are currently working on extending this questionnaire into a formal assessment rubric. We envision that this tool will help instructors effectively evaluate a potential project using a quantitative metric spanning multiple criteria.

ACKNOWLEDGMENTS

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

REFERENCES

[1] [n. d.]. Senior Project, Department of Software Engineering, Rochester Institute of Technology. https://seniorproject.se.rit.edu/. Accessed: 2024-01-12.

[2] Aaron Bloomfield, Mark Sherriff, and Kara Williams. 2014. A service learning practicum capstone. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education* (Atlanta, Georgia, USA) *(SIGCSE '14)*. Association for Computing Machinery, New York, NY, USA, 265–270. https://doi.org/10.1145/2538862.2538974

[3] Grant Braught, John Maccormick, James Bowring, Quinn Burke, Barbara Cutler, David Goldschmidt, Mukkai Krishnamoorthy, Wesley Turner, Steven Huss-Lederman, Bonnie Mackellar, and Allen Tucker. 2018. A Multi-Institutional Perspective on H/FOSS Projects in the Computing Curriculum. *ACM Trans. Comput. Educ.* 18, 2, Article 7 (jul 2018), 31 pages. https://doi.org/10.1145/3145476

[4] Bernd Bruegge, Stephan Krusche, and Lukas Alperowitz. 2015. Software Engineering Project Courses with Industrial Clients. *ACM Trans. Comput. Educ.* 15, 4, Article 17 (dec 2015), 31 pages. https://doi.org/10.1145/2732155

[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] Janet Davis and Samuel A. Rebelsky. 2019. 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* (Minneapolis, MN, USA) *(SIGCSE '19)*. Association for Computing Machinery, New York, NY, USA, 29–35. https://doi.org/10.1145/3287324.3287508

[7] Pierre Flener. 2006. Realism in Project-Based Software Engineering Courses: Rewards, Risks, and Recommendations. In *Computer and Information Sciences – ISCIS 2006*, Albert Levi, Erkay Savaş, Hüsnü Yenigün, Selim Balcısoy, and Yücel Saygın (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1031–1039.

[8] Vahid Garousi. 2010. Applying Peer Reviews in Software Engineering Education: An Experiment and Lessons Learned. *IEEE Transactions on Education* 53, 2 (2010), 182–193. https://doi.org/10.1109/TE.2008.2010994

[9] Vahid Garousi, Gorkem Giray, Eray Tuzun, Cagatay Catal, and Michael Felderer. 2020. Closing the Gap Between Software Engineering Education and Industrial Needs. *IEEE Software* 37, 2 (2020), 68–77. https://doi.org/10.1109/MS.2018.2880823

[10] 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

[11] Nicole Herbert. 2018. Reflections on 17 Years of ICT Capstone Project Coordination: Effective Strategies for Managing Clients, Teams and Assessment. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (Baltimore, Maryland, USA) *(SIGCSE '18)*. Association for Computing Machinery, New York, NY, USA, 215–220. https://doi.org/10.1145/3159450.3159584

[12] Birgit Krogstie and Bendik Bygstad. 2007. Cross-Community Collaboration and Learning in Customer-Driven Software Engineering Student Projects. In *20th Conference on Software Engineering Education & Training (CSEET'07)*. 336–343. https://doi.org/10.1109/CSEET.2007.17

[13] Amruth N Kumar and Rajendra K Raj. 2022. Computer Science Curricula 2023 (CS2023) Community Engagement by the ACM/IEEE-CS/AAAI Joint Task Force. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 2*. 1212–1213.

[14] Stan Kurkovsky. 2022. Using Scaffolding to Simplify FOSS Adoption. In *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 2* (Dublin, Ireland) *(ITiCSE '22)*. Association for Computing Machinery, New York, NY, USA, 587–588. https://doi.org/10.1145/3502717.3532163

[15] Stan Kurkovsky. 2023. 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 Vol 1* (Hyderabad, India) *(CompEd 2023)*. Association for Computing Machinery, New York, NY, USA, 112–118. https://doi.org/10.1145/3576882.3617929

[16] Joseph Mertz and Scott McElfresh. 2010. Teaching Communication, Leadership, and the Social Context of Computing via a Consulting Course. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education* (Milwaukee, Wisconsin, USA) *(SIGCSE '10)*. Association for Computing Machinery, New York, NY, USA, 77–81. https://doi.org/10.1145/1734263.1734291

[17] Vito Moreno, Eric Cutiongco, and Vinay Patel. 2016. Development of an Industrially Sponsored Senior Design Capstone Program: A University and Sponsor Perspective. In *ASME International Mechanical Engineering Congress and Exposition*, Vol. 50571. American Society of Mechanical Engineers, V005T06A034.

[18] Andres Neyem, Jose I. Benedetto, and Andres F. Chacon. 2014. Improving software engineering education through an empirical approach: lessons learned from capstone teaching experiences *(SIGCSE '14)*. Association for Computing Machinery, New York, NY, USA, 391–396. https://doi.org/10.1145/2538862.2538920

[19] 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

[20] Gustavo Pinto, Clarice Ferreira, Cleice Souza, Igor Steinmacher, and Paulo Meirelles. 2019. Training Software Engineers Using Open-Source Software: The Students' Perspective. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. 147–157. https://doi.org/10.1109/ICSE-SEET.2019.00024

[21] Rajendra Raj, Mihaela Sabin, John Impagliazzo, David Bowers, Mats Daniels, Felienne Hermans, Natalie Kiesler, Amruth N. Kumar, Bonnie MacKellar, Renée McCauley, Syed Waqar Nabi, and Michael Oudshoorn. 2022. Professional Competencies in Computing Education: Pedagogies and Assessment. In *Proceedings of the 2021 Working Group Reports on Innovation and Technology in Computer Science Education* (Virtual Event, Germany) *(ITiCSE-WGR '21)*. Association for Computing Machinery, New York, NY, USA, 133–161. https://doi.org/10.1145/3502870.3506570

[22] Thomas J. Reichlmayr. 2006. Collaborating with Industry: Strategies for an Undergraduate Software Engineering Program. In *Proceedings of the 2006 International Workshop on Summit on Software Engineering Education* (Shanghai, China) *(SSEE '06)*. Association for Computing Machinery, New York, NY, USA, 13–16. https://doi.org/10.1145/1137842.1137848

[23] Jan-Philipp Steghöfer, Håkan Burden, Regina Hebig, Gul Calikli, Robert Feldt, Imed Hammouda, Jennifer Horkoff, Eric Knauss, and Grischa Liebel. 2018. Involving External Stakeholders in Project Courses. *ACM Trans. Comput. Educ.* 18, 2, Article 8 (jul 2018), 32 pages. https://doi.org/10.1145/3152098

[24] Kathleen Timmerman and Michael Goldweber. 2022. Department-wide Multi-semester Community Engaged Learning Initiative to Overcome Common Barriers to Service-Learning Implementation. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education - Volume 1* (Providence, RI, USA) *(SIGCSE 2022)*. Association for Computing Machinery, New York, NY, USA, 808–814. https://doi.org/10.1145/3478431.3499326

[25] Jari Vanhanen, Timo O. A. Lehtinen, and Casper Lassenius. 2012. Teaching real-world software engineering through a capstone project course with industrial customers. In *2012 First International Workshop on Software Engineering Education Based on Real-World Experiences (EduRex)*. 29–32. https://doi.org/10.1109/EduRex.2012.6225702