© Journal of Higher Education Outreach and Engagement, Volume 26, Number 1, p. 129, (2022)
 eISSN 2164-8212

A Case Study of a Multiyear Community-Engaged Learning Capstone in Computer Science

Amy Csizmar Dalal and Emily Oliver

Abstract

We present a case study of a multi-year, academic civic engagement (ACE) collaboration in a computer science capstone course. ACE projects in computer science provide an avenue for students to apply software development concepts to real-world projects with actual clients, and can offer meaningful engagement with ethical issues. The typical time-limited nature of ACE projects within a single course leaves little time for reflection, iterative development, maintenance, and evolution of priorities, centering student learning outcomes over community partner goals. The model presented here is sustainable and robust to changes in personnel on both the community side and the academic side, including student participants. We highlight the importance of an involved center for civic engagement to facilitate relationship formation and frame civic learning for students. We address how longevity facilitates a true iterative and collaborative development process, supports the development of trust relationships, and opens up space for transformational change.

Keywords: academic civic engagement; case study; collaborative software development; computing for good

Academic civic engagement (ACE) projects have long enabled students to apply curricular concepts to real-world projects with a community service benefit. Such projects pair students in a course, or a capstone experience, with one or more community organizations. In the best-case scenario, the students and the community organization collaborate on a project of joint interest that serves as a mechanism to apply learning to real-world problems for students and results in a tangible outcome or product for the community partner.

Within computer science, ACE projects provide an avenue for students to apply software development concepts to real-world projects with actual clients. Designing and implementing real software for real people forces students to confront user-centered and algorithmic design issues that are easy to ignore in class projects. Such experiences provide professionalism practice for students who are likely to work in similar scenarios once they complete their degrees—practice that is difficult to replicate in a classroom environment. ACE projects also can offer meaningful engagement with ethical issues in a way that classroom readings, discussions, and simulations cannot.

In well designed and executed ACE projects in computer science (ACE in CS), community partners benefit from the technical assistance that student projects provide. Often, money for IT, software procurement, and/or software development projects is tight, nonexistent, or better spent in other areas of the nonprofit. Outsourcing these tasks to computer science students ideally saves the community partner the expense of a commercial solution and/or saves time spent researching various alternatives, time better spent in the core work of the nonprofit. Additionally, student–community partnerships expose students to the work and goals of the nonprofit organization and provide students a broader context of the needs and daily workings of the surrounding community.

On balance, such partnerships tend toward more favorable outcomes for the students than for the community partner, due largely to the projects' structure and nature (Mitchell, 2008). Chief among the limitations: time. Projects tend to last for a semester or quarter, leaving little time for reflection, iterative development, or meaningful progress. Because students fit in ACE work with their other coursework, their commitment to the project is part time. Software rarely works correctly 100% of the time, and requires bug fixes and updates over time. The needs of the organization may change, rendering the project obsolete, often sooner than either party expects. A short-term collaboration addresses none of these issues, forcing partners to either abandon the solution altogether or spend time and money to fix the issues on their own. Increasingly, both sides are questioning the ethics of this particular model of "drop-in" collaboration, from both the community partner's perspective and the curricular perspective.

One solution is to create multiyear collaborations between community partners and an evolving group of students, either over several offerings of the same course or, as we discuss here, in capstone projects spanning several years. Such longer lived collaborations address some of the issues around maintenance, iterative development, testing, and morphing of goals and priorities, as well as some of the ethical issues. Executed well, such a model has the potential to strengthen community–academy relationships, specifically allowing for the development of deeper trust relationships. It may also provide a stronger model for ethical software development for computer science students, addressing many of the ethical issues outlined above.

This case study presents a model of a sustainable and sustained collaboration between community members and the academy that is robust to changes in personnel on both the community side and the academic side. Our case study demonstrates effective ways to onboard new project members from both the community side and the academic side, lessons we learned from trial and error. We highlight how aspects of an iterative software development process facilitate the community–academy feedback process and enrich the development process on both ends.

The case study also describes lessons learned in the course of this partnership, lessons about developing trust between the parties and about developing respect within the students (and faculty) for the lived experiences and expertise of the community partners. It highlights the importance of an involved center for civic engagement to facilitate relationship formation and frame the academic and social aspects of the work for the students—as well as providing space for necessary and fruitful reflection by students on their learning, positionality, and experiences. It describes some of the unexpected mundane details that have proven important, such as producing documentation. Finally, it presents a mechanism for project maintenance and growth once the formal academic partnership ends.

Situating Civic Engagement in a Computer Science Context

The literature situating civic engagement, sometimes called "service-learning" within the broader academic context, is well established. Reviewing a range of existing literature, Mitchell (2008) explored the divide in service-learning between a traditional approach that emphasizes course-based service without attention to the structural nature of inequity on one hand, and a critical approach that explicitly seeks to challenge the systems of injustice on the other. She highlighted a social change orientation, working to distribute power, and developing authentic relationships, as characteristics that distinguish a critical service-learning project. She contested that the goals of student development (preprofessional experience, leadership skills, etc.) and community change are mutually exclusive, suggesting that focusing on community partners' goals will also lead to positive outcomes for students. Although the Center for Community and Civic Engagement at our institution (Carleton College) uses the terminology of academic civic engagement, Mitchell's conceptualization of critical service-learning certainly echoes the CCCE's student learning objectives:

- Understanding issues in their real world complexity.
- Recognizing and honoring different forms of knowledge.
- Awareness of positionality.
- Doing—how can students take the course content and do something

with it beyond the classroom while learning in the process.

- Developing leadership skills.
- Nurturing a commitment to lifelong civic engagement. (Center for Community and Civic Engagement, n.d., Vision Statement)

Mitchell's discussion of critical service-learning also resonates with the aspirational best practices the Center hopes to promote, via workshops, training, and events, among Carleton faculty.

Whitney et al. (2016) discussed the interaction and tension between academic goals and community needs. With the aim of understanding and addressing future opportunities and challenges for reforming service-learning and community engagement in higher education, they examined the on-the-ground efforts of two community organizations to illuminate some of the recurrent issues associated with democratic engagement. The coauthors—a combination of academics and leaders or staff of the two organizations—highlighted several areas of tensions, namely asset-oriented norms and cocreation, place-based partnerships, and a process orientation toward impact. The organizations' work illustrates the complexities of democratic engagement, which can sometimes be exacerbated by partnerships involving the academy, especially with an orientation primarily toward student learning outcomes (Trebil-Smith, 2019).

The field of civic engagement at large has paid increasing attention to the serious risks of one-time, transactional student–community engagement. The work of Susan Gust, a community organizer, and Catherine Jordan, an academic, reflects the process of recognizing and working through such risks in the long-running partnership between community activists and the University of Minnesota that led the Phillips Neighborhood Healthy Housing Collaborative (PNHHC), a group of local residents, to make transformative change in the community, the university, and the lives of the participants (Jordan & Gust, 2011). Gust and Jordan described their own disparate backgrounds as collaborators, explicitly naming the self-interest that led them to become involved in the project, and recounted how the challenges of learning to work equitably across lines of difference through the PNHHC affected their collaborative practice and provided profound personal benefits. This work has led them to develop a community impact process for potential higher education–community collaborations (Gust & Jordan, 2006). Another vital example comes from scholars Katie Johnson-Goodstar and Jenni Sethi, who worked in collaboration with attendees of a 2014 presentation to create the "But Do *I* Want to Work With *You*" checklist to support community organizations in having more agency to decline proposed collaborations with institutions of higher education that do not align with their values or advance their goals (Johnston-Goodstar & Sethi, 2014).

Literature over the past few decades has explored placing, or centering, civic engagement projects within computer science courses. Similar to Mitchell, Connolly (2012) questioned the prevailing approach of "service-learning as internship" in computer science, with outsized benefits to the students at the expense of the community partners, and argued for an "advocacy orientation" to service-learning instead. Perhaps the most similar model to the one we discuss here is the software design course discussed in Davis and Rebelsky (2019). Students in this course developed non–mission critical software for local nonprofit organizations, following a more traditional client/contractor structure than the cocreator structure we describe here. Teams in one semester hand off code to teams in subsequent semesters. Vennekens (2020) presented a small case study of a web technology course where students partnered with a single nonprofit organization to develop games for the organization's platform. The partnership's main goal was to increase student engagement in the course and develop a greater sense of student empathy, which fits Mitchell's model of traditional service-learning.

Dekhane et al. (2018) described an elective course where students designed outreach activities to introduce children and nonmajor first-year students to computing, with a focus on retention of minoritized students in computing through service-learning. Sabie and Parikh (2019) reported on a master's-level service-learning course partnering students with nonprofit organizations to work on open-ended problems. The course prioritized the development of relationships of care over the creation of artifacts and finished products, and focused

on the act of cocreation between students and community partners; the collaborations end when the course ends, with no carryover to subsequent course offerings. Syeda et al. (2020) introduced a framework for integrating service-learning design studies into semester-long data visualization courses. The service-learning model in this course fits Mitchell's (2008) definition of traditional service-learning, focusing almost exclusively on student learning outcomes. Although not part of a classic course per se, Lee et al. (2019) described a MOOC-like environment where students contributed to the development of websites for nonprofit organizations through "microroles." Although the microroles allowed students to collaborate on increasingly complex tasks, the structure precluded true cocreation between the nonprofit organization and the learners.

Humanitarian free and open source software (HFOSS) projects (Parra et al., 2016) overlap the service-learning space: the open source nature means anyone can contribute, and the humanitarian aspect means the software development project focuses on fostering some social good. Because students take time to learn the norms of the project and the project's developer community, such projects are well-suited for multiterm courses such as capstones. Braught et al. (2018) reported on five different models for capstones engaging students in HFOSS projects, some of which, like the project reported on here, lasted over multiple semesters. HFOSS projects share the cocreation structure of transformational service-learning projects, but do not necessarily tie students to their local communities, as the projects may literally be hosted all over the world. In addition, many HFOSS-based courses prioritize knowledge about software development workflows and tools in open source software development over cocreation of knowledge.

The ethics of working with community partners and nonprofit organizations through ICTD (Information and Communications Technology for Development) research is a commonly explored theme in literature from the computer science subfield human–computer interaction (HCI). Bopp and Voida (2020) presented an important overview of the space, delving into the biases inherent in existing research in terms of types of organizations represented, types of methodologies, which stakeholders are given "voice," and so on. Dell and Kumar (2016) critically examined the field of HCI for development (HCI4D) via literature review and interviews with domain experts, concentrating on understanding the current landscape and prevailing attitudes about what HCI4D is and what role it plays in HCI. Voida (2011) outlined the challenges inherent in working with nonprofit organizations, particularly as their resources, goals, and operations shift in response to events in the public, private, and household/community sectors. Value sensitive design (VSD) is often used as a framework for approaching ethical research with community partners; Borning and Muller (2012) discussed the limits of VSD as traditionally practiced, and provided suggestions for addressing issues of defining values, giving voice to stakeholders, and so on. Similarly, Dombrowski et al. (2016) described a social justice orientation for research addressing large-scale social issues, focusing on six dimensions—transformation, recognition, reciprocity, enablement, distribution, and accountability—and three commitments—to conflict, to reflexivity, and to personal ethics and politics. These works resonate with the themes in Mitchell (2008) as well as the learning objectives of Carleton's CCCE office.

Project Structure

Academic civic engagement or service-learning has long been discussed among the "high impact practices" that leave lasting imprints through active student learning. Building off Kuh's (2008) work on high impact practices, which in addition to civic engagement include academic capstone and undergraduate research experiences, attention has increasingly focused on the particular benefits for students from underrepresented groups in higher education: BIPOC, first-generation college, transfer, and low-income students. Finley and McNair (2013) noted the "equity effects" of high impact practices because, while they influence learning across groups of college students, their impact appears to be *more* significant for students from groups who are historically and currently underserved by higher education. Additionally, Finley and McNair's analysis bolsters the claim that participation in *multiple* high impact practices over the course of a college experience can influence self-perceptions of learning, particularly for students from underserved groups. The case study explored in this article, an academic

civic engagement project embedded across a two-trimester computer science capstone, is an example of multiple, simultaneous high impact practices.

Academic civic engagement provides vital space for students and faculty to grapple with the ethical dimensions and potential public purposes of their fields. The resulting class-inspired discussions and reflections are microcosms of the larger conversations about the public purpose of higher education. In their seminal white paper, Saltmarsh et al. (2009) envisioned this purpose as the site of reciprocal collaboration to aid in "public problem solving." Because access to technical knowledge of computer science is often limited in the small nonprofits or grassroots organizations with which the Carleton courses often collaborate, computer science has an especially potent potential to expand the capacity of people doing transformative work in communities through public problem solving. By the same token, computer science collaborations present unique challenges in achieving the "full participation" of community collaborators (Strum et al., 2011) and require a heightened attention to communication, positionality, trust building, and agenda cosetting.

The long-term and iterative structure of the project featured here provided more space for community partner participation and revision than a typical, single-term ACE project. As a collaborative capstone project, it is also the culmination of an informal "pathway of civic learning," which along with various recurrent and one-time computer science, math and statistics, and physics ACE courses, showcases avenues for applying students' technical STEM skills for the public good. As we discuss in greater depth later, this particular project structure, within the long-established framework of academic civic engagement, provides powerful benefits to student learning, the community partner relationship, and the actual impact of the product of the collaboration on the community partner's workflow.

Senior Comprehensive Exercise "Comps"

Carleton College mandates a capstone experience, "Comps" (short for senior comprehensive exercise), in the major for every student, typically completed in the student's senior year or last year at Carleton. It is a cultural norm at the college that students take their Comps project seriously, putting significant academic as well as emotional weight on "Compsing."

Computer science's Comps spans two consecutive trimesters of an academic year, counting as half of a course credit in each term. Computer science majors work in teams of four to six students, assigned by the department, on a project chosen by their faculty advisor that engages some subset of their major coursework. Most commonly, students draw heavily on Algorithms and Software Design, two required courses in the major. Increasingly, projects rely on some student knowledge of artificial intelligence, machine learning, statistics, data visualization, and/or HCI. Besides practicing effective teamwork strategies, a valuable life skill and career skill in the software development industry, students also practice using the tools of the trade to manage code repositories, conduct code reviews, and keep track of milestones and work in progress. Projects range from more traditional software development projects, a subset of which are performed with community or campus partners as ACE projects, to more academic projects, such as conducting research or analyzing algorithms. Students take ownership over the ill-structured problems, with light guidance from faculty.

In the first term, students immerse themselves in the problem space. In software development projects, they conduct research into the audience and goals and develop user stories. The group produces a project proposal, which includes a timeline of milestones and deliverables, along with artifacts like architectural diagrams, a literature review, and an algorithm outline. By the end of the term, the team completes an alpha version of their solution based on the project proposal.

In the second term, students refine and complete their solution. They present their work publicly at a Comps symposium; the community partner attends if they are able. At the conclusion of the project, they release source code or other artifacts and publish their results on a website hosted by the department. For projects that are likely to continue in a subsequent year, students produce handoff documentation for the next team.

Identifying and Building Relationships With Community Partners

Two mechanisms exist to match community

partners with courses. In one model, the campus Civic Engagement office acts as a clearinghouse, identifying and vetting community partners and connecting interested faculty. This model places the burden on the Civic Engagement office to develop relationships with community partners and faculty independently, identify potential fruitful connections, and identify potential faculty/course fits for a particular community need. The advantage of this model is that information about work between the community and the campus is centralized, giving the Civic Engagement office the most complete knowledge of the number and nature of community/academic connections.

In another model, faculty develop relationships with community partners independently of the Civic Engagement office, looping in the Civic Engagement office once the partnership is established. This model places the burden on faculty and community partners alike to identify and build upon potential curricular connections. In a new partnership between the community partner and the campus, the onus is on the partner to vet the faculty member, and on the faculty member to assess the suitability of the match. Of course, the Civic Engagement office, once looped in, can perform or at least assist with these tasks, given the strength of their community knowledge overall. However, it also recognizes and takes advantage of the relationships that serendipitously arise when faculty and community members meet and connect in any number of contexts.

The faculty–community partnership described here began serendipitously via a student connection. The student attended a panel of community organizations hosted by the Civic Engagement office, where they heard the community partner describe their need to keep better track of the youth utilizing their services. The student connected with the community partner after the panel, simultaneously mentioning the encounter to the faculty member and asking if this could form the foundation of a computer science Comps project.

Relationship building proceeded on several levels. The faculty member met with the community partner and the student to create a project outline. The faculty member and community partner codeveloped a Comps project proposal for the following academic year based on this outline, with the goal of moving the partner from paper-based attendance tracking to electronic attendance tracking. The faculty member looped in the Civic Engagement office to designate the project as an "ACE course" and acquire necessary course support. The faculty member and community partner met several times prior to the start of the project to discuss project and support details and to clarify expected outcomes. By the time the project started, a process and structure were already in place to support the students.

The faculty member leveraged preexisting strong relationships with the Civic Engagement office forged through previous course and capstone civic engagement projects. The Civic Engagement office was already well-versed in the faculty member's interests and strengths, and knew what the faculty member would bring to the partnership. The Civic Engagement office also knew, based on past experience, that the faculty member would be an appropriate match for this community partner. The Civic Engagement office thus provided valuable vetting to the project, a critical factor in the project's success. Additionally, the student was both primed to reflect on how their computer science major could be utilized to facilitate community change, and empowered to initiate connections with community members independently. This is a key example of student-directed pedagogy at work and a clear demonstration of civic agency (Boyte, 2009).

These early meetings between the community partner and the faculty member are crucial for building trust between the two, and for managing expectations. The faculty member needs to be honest about what students can and cannot bring to the partnership. It's also helpful if the faculty member can anticipate potential pitfalls that may affect the project's progress and/or deliverables, and work with the community partner to develop a contingency plan. Being honest about outcomes, and then delivering on those outcomes to the extent possible, facilitates and expands trust between the two parties.

Having clear expectations up front helps the community partner fit the project deliverables and timeline into their important community work. Taking the worry about the project off their plate, to the extent possible, allows them to concentrate on their core work. The relationship with campus should be a benefit, not a burden, and the

faculty member, in addition to the civic engagement center staff, needs to play a major role in making this so.

About the Community Partners

The Key is the oldest youth-run youth center in the nation. It is run by Northfield Union of Youth Key youth board, which is democratically elected by youth. They hire and review all staff as well as make programming and policy decisions about The Key.

A community partner's commitment to participatory, democratic engagement is an asset to an academic civic engagement collaboration. First, an organization such as this one lends itself particularly well to what Mitchell (2008) defined as "critical service-learning" pedagogy, where students are encouraged "to see themselves as agents of social change, and use the experience of service to address and respond to injustice in communities" (p. 51). Again, this approach seeks to counter the long history of paternalism in university–community partner relationships, urging faculty to incorporate ideas about systems of power into the courses, as opposed to "traditional" service-learning's focus on only direct service. Because youth self-determination and systemic issues around equity and access are fundamental to the work of The Key, the computer science student collaborators are compelled to design a tool with those concepts in mind. The Key staff too, because of their organization's culture and values, are also adept at naming and managing relational power dynamics, which can support effective communication between collaborators. Lastly, when students see mission-driven organizations in action, through site observations and active collaboration, they are able to gain a greater sense of the potential impact of their project, which can inspire deeper student investment.

The Key has an extensive history of collaboration with Carleton's Center for Community and Civic Engagement, regularly partnering on several academic civic engagement projects a year. This frequency has established a level of trust and has even shaped some overlapping philosophies around collaborations. Trebil-Smith (2019) is among the scholars of civic engagement who have noted that a solid foundation of collaboration is often an element of successful civic engagement projects, especially around community partners having space for an expansive vision of potential longer-term outcomes. "For those with more established partnerships, the vision tended to include long-term, sustainable programs and full-circle, student-led initiatives (i.e. students designing, implementing, and sustaining a project or program)" (p. 21).

The Healthy Community Initiative (HCI) joined the collaboration in the second year of the project. Like Carleton College, HCI is located in Northfield, Minnesota; it self-defines its mission as "cultivat[ing] a collaborative community that supports, values and empowers youth" (HCI, 2020, We Support Local Youth Programs). In addition to its own in-house programming, it frequently serves as a convener for stakeholders invested in youth empowerment in Northfield and, increasingly, in surrounding Rice County. The organization also coordinates relevant efforts, and because of its successful grantmaking, plugs in staff resources or available funding to bolster the work of partners on shared priorities. HCI became involved in this project because The Key and HCI routinely share data in order to identify and allocate resources to youth within the community. HCI thus had an interest in what data was collected, and how this data could be shared with them.

Similar to The Key, HCI has a long-standing relationship with Carleton's CCCE. The HCI director is a College alum and has served as a community partner representative on the CCCE's oversight committee. Having a project that, as it develops momentum, involves additional community partners is also a way to showcase to students that, for the goals of a community change agenda to be met, the effort often needs to include various stakeholders. For example, the project eventually incorporated attendance data from the high school so that The Key's staff could be more agile in identifying youth in crisis.

Project Lifecycle

Multiyear projects such as this one require attention to multiple timelines: the day-to-day structure of a single Comps cycle, as well as the between-cycles planning and reflection. In addition, the nature of this particular collaboration required special considerations around data privacy and confidentiality.

Structure of a Single Comps Cycle

An individual Comps cycle begins with a kickoff meeting, where the student team and the community partner review and codevelop goals and deliverables for this cycle. The students hear firsthand from the partner about what's been working well, what's not working at all, and other problems or issues with the current system. Because the CCCE and The Key have an established relationship, The Key's leadership staff are practiced in this cocreation process, and thus take both a leadership and a mentorship role as the students navigate this process for the first time. The community partner sets the agenda and shares ownership of the cocreated goals, resetting the typical power structures as discussed in Mitchell (2008).

Students then meet as a team without the community partner to conduct their own review of goals and deliverables. They review notes from the previous cycle, if applicable, including the list of unimplemented deliverables and features, prioritizing the ones the community member highlighted as important. They develop a plan to review the existing codebase.

Site observations are an especially important element in this process, and occur early in the cycle. Through observation, students get a much clearer picture of what it looks like for The Key to deliver its services and live out its mission. They see for themselves the strengths and limitations of the existing workflow. Although early discussions and meetings are fruitful, the group's focus and temperament change after these observations. We discuss the benefits and challenges of observation as a research method in the section Discussion and Lessons Learned.

The team meets at least once a week with the faculty advisor to review their progress and to hash out design or technical issues. Team members meet on their own several times between faculty advisor meetings, either as coworking sessions or for further discussion of technical and design issues.

The team meets at least once with the community partner during each term, although ideally these meetings occur on a more regular basis. During the second year of the project, for instance, the team met every other week with the community partner. At these meetings, the partner and team review and refine goals and deliverables, and the team demonstrates the latest progress. The meetings help to keep the team on track and accountable to the partner, and remind the team to center the partner's agenda. They also help prevent "drift," where the actual development deviates from the partner's goals and needs.

To ensure the system would run robustly when deployed, the students conducted both usability tests and soft deployments. Students conducted the former during meetings with community partners, to get one-off feedback on, say, the placement of buttons and fields or the understandability of labels and functionality. During soft deployments, students monitored the database to verify that records remained stable and updated properly. They stress-tested the system to confirm it could handle peak loads. Volunteers and staff at The Key provided valuable feedback on how to streamline data entry and on bugs that popped up while in use.

In the term following the completion of Comps, students meet with the partner one last time for an official "handoff" and release of the production version of the software.

Between Comps Cycles

At the conclusion of a Comps cycle, the advisor and community partners debrief, without students present. The meeting focuses on practical questions: What went well in this partnership? (How) are you using the software? What are the main issues you are encountering with the software? Should we continue this partnership next year? Having this established space for honest community partner feedback at the end of a cycle of working together is an important equity practice that acknowledges the power dynamics a faculty person can bring into a collaboration.

The decision to continue is largely based on the goals that the software is not meeting, or not fully meeting. In the original conception of the project, one of the long-term goals stated by the community partner was the ability to demonstrate to donors, grantors, and potential donors and grantors, the effectiveness of The Key's programs, using actual data. Our year-over-year decisions have largely hinged on whether continuing the project for another year would move The Key closer to this goal. This decision is balanced on the academic side by asking, Would students' work in continuing the

project meet the learning goals of computer science Comps? If the project instead seems chiefly an exercise in maintenance, it would not continue as a Comps project for the next year.

Once the decision is made to continue, the community partner and faculty member set goals and objectives for the next Comps cycle. This iterative codevelopment of objectives and deliverables is crucial to the continued success of the project. It honors and centers the community partner's knowledge and experience, integrating it holistically into the learning objectives of Comps, so that the needs of both sides are met to the extent possible (Jordan & Gust, 2011).

Finally, the faculty member facilitates the onboarding process for the new project team. The incoming project team meets with the outgoing project team in late spring the year before the next cycle, once the teams and projects for the next year are established. The outgoing team shares accomplishments, known issues, and next steps. The incoming team peppers them with questions about the project. The outgoing team provides access to the code repository, along with any other information necessary (Amazon Web Services keys, etc.) for getting started with the codebase.

Special Considerations

The clientele of The Key consists largely of minor children, some of whom fall into additional underserved groups: they are unsheltered or housing insecure, food insecure, and so on. This meant we needed to take extra care with data privacy, ensuring, to the extent possible, that data was available only to certain parties on a need-to-know basis, while still allowing staff members, volunteers, and youth the ability to take attendance. The addition of HCI to the project, and the ensuing integration of school-related data, lent an additional importance to data privacy considerations. The data privacy issues were most salient when structuring the reporting functionality and some aspects of the sign-in functionality.

Results

Figure 1 shows the progression of the site development over the span of the project and the evolution of project goals. The site progression summarizes the core work in each year of the project: the foundational work in Year 1, and the iterative refinement of both the vision and the implementation in Year 2 and Year 3.

Year 1: "Throw One Away"

There is a saying among software developers that the first version of any product you develop is the one you throw away. This is the version where you figure out what the problem actually is as you are trying to solve it, and where you make the majority of your design mistakes. The saying acknowledges that software developers, like writers, need

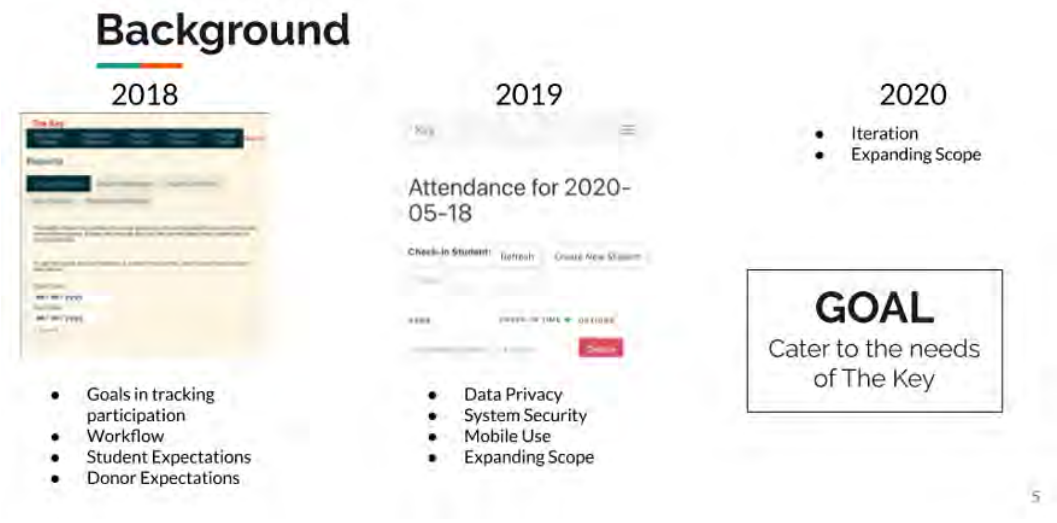

Figure 1. Summary of Deliverables Year Over Year
Note. Slide generated by the Year 3 Comps team.

the rough draft to figure out exactly what they want to say and how to say it. In the first year of the project, the students wrote the version of the software that we threw away.

Throwing out what we did was of course not the goal of the project. At that point, it was not even clear that the project would expand beyond the first year. As far as the students were concerned, they were writing *the* version of the software that would be used moving forward.

The major goal of the project in Year 1 was to move The Key from paper-based attendance tracking to electronic attendance tracking via a database-driven website. The system modeled attendance as "one sheet per day," based on the team's observations of the volunteers' workflow at The Key. The Comps team wanted their system to mimic the paper-based workflow as much as possible while providing vital enhancements, to avoid cognitive dissonance and the stress of learning a new workflow.

The website (Figure 2) mimics a spreadsheet with multiple tabs representing multiple views of the data. Entering student names is front and center, in the first (default) tab. From this tab, users can also view and download past attendance "sheets." The Attendance Overview tab provides an ability to download attendance sheets within a date range for offline processing, useful when generating reports for grant agencies. The Student Profiles tab allows volunteers to view and edit information about students. The Attendance Columns tab allows The Key's leadership to add and modify the programs and activities tracked over time—a need identified in the course of codevelopment. The Reports tab tracks how many unique students participated in a programming category, total student attendance by date range, activity participation by date range, new attendees by date range, and other attendance milestones. All of these features were either noted as important during the observation phase or indicated as important during the requirements-gathering phase.

The final version fulfilled most of the requirements, but left others incomplete. "Manage Profile," an attempt to merge multiple records of the same person (for example, under different names and nicknames), never completely worked, and the team was unable to implement uploading student pictures to their student profiles. The site proved unstable, performing differently on different web browsers and occasionally losing data. The team designed and implemented the site to work optimally on a desktop or laptop, yet the volunteers used mobile devices to record attendance—a fatal flaw that quickly became evident to the team at the site's soft rollout. In addition, the site was not secure: None of the actions required a login, which meant anyone had

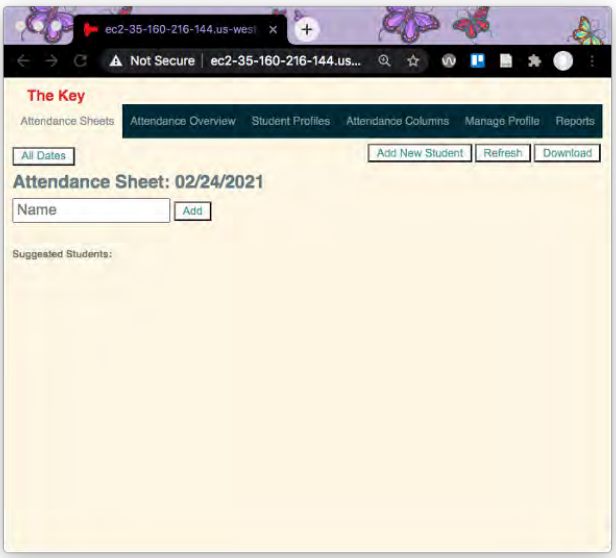

Figure 2. Screenshot of the Main Tab Showing the Attendance Entry Page in the Year 1 Prototype.

access to any of the data within the system.

In Year 2, we had to start over.

Year 2: Revamping the Model

The major goals of the 2nd year of the project were to fix the security issues in the original website and to improve the mobile experience. To accomplish these goals, the team redesigned the site from the ground up. They reimplemented all of the previous year's features and redid the entire database to make it more robust. They added basic authentication, requiring users to login before performing any operation, albeit with a single sign-on username and password for all volunteers and leaders, with no differentiation between roles.

The team modified some of the reporting capabilities of the site, allowing some online analyses and "heat map" visualizations, as Figure 3 shows. The Reports tab retained the ability to download data for offline analysis. In practice, the visualizations proved a little too clunky for The Key's purposes.

Although the site was a major improvement over the previous year's offering, issues remained. The lack of differentiated roles left minors' data exposed to anyone with login credentials, a violation of the system's data privacy requirements. The system documentation was also lacking, which made it hard for the Year 3 students to get up to speed, and for The Key's leadership to figure out why certain bugs occurred.

Year 3: "Putting Out Fires"

The Year 3 team faced two significant challenges: a switch in faculty advisors from the first 2 years of the project, and the arrival of the global COVID-19 pandemic mid-project. The advisor was new to the project and new to Comps advising, and grappled with both the complexity of the project and with learning how to effectively advise Comps. The pandemic moved Carleton immediately from in-person learning to remote learning, requiring the team to figure out how to work together on the codebase remotely for the entirety of the second term of the project. The pandemic also shut down The Key to all in-person programming, which would impact the team's ability to test and roll out any changes to the codebase—a point we return to later in this section.

This cycle's work expanded the scope of the site to assess and articulate program out-

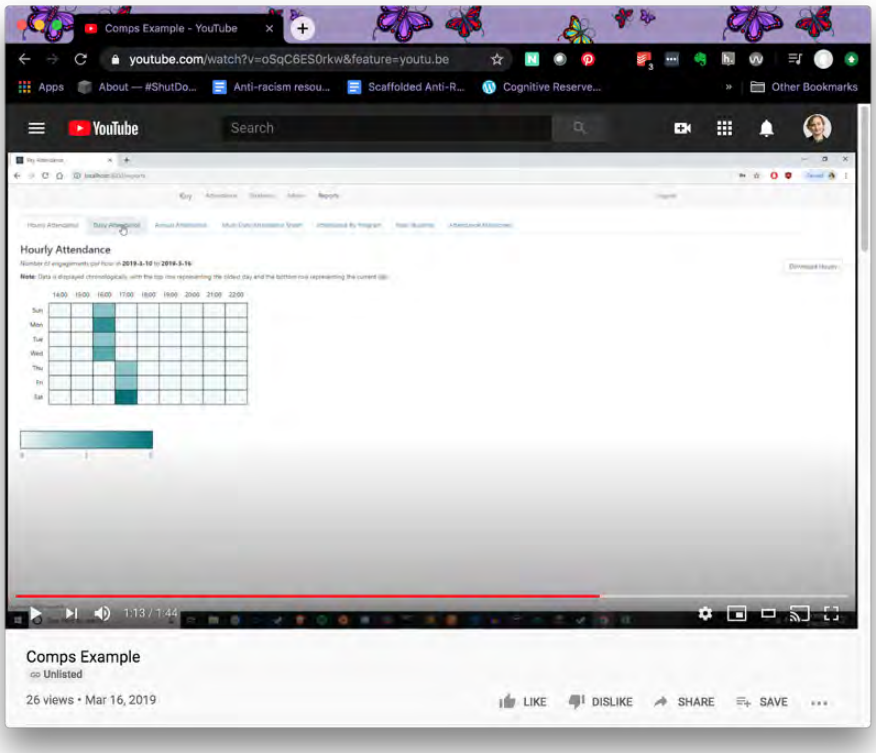

Figure 3. Screenshot of the Reports Tab in the Year 2 Prototype

comes and effectiveness for The Key staff and donors, including a mechanism to track volunteers and a "flag" system to track student needs such as food, housing, mental health, and employment. The team implemented the ability to search by student key (ID number), providing another way to connect multiple profiles for the same student and to connect The Key's information with school data.

The team struggled to make sense of the codebase, even with the assistance of a mentor from the Year 2 team. Eventually, they paused development to create better documentation for the codebase, and to verify that they could integrate a small change to the existing codebase. Although producing documentation and integrating small changes to the code at first is a strategy we have used when advising Comps projects that contribute to open source codebases, we had not thought to apply it in this context. This process uncovered structural and security issues with the code that needed to be addressed immediately, which took priority over other development tasks.

The team improved site security by implementing user roles and multiple logins, addressing the issues with information sharing of data associated with minor children. They fixed various software bugs and resolved a number of code dependencies stemming from outdated packages. They cleaned up the interface to address some of the usability issues that arose in day-to-day use.

These deliverables were absolutely necessary for the code to remain viable, but the team's contributions felt less like the fundamental system design of previous years' work. The team spent more time reacting to the needs of the project than to proactively advancing a design vision. Although managing these practical details was absolutely necessary for this part of the project, the project had a different "feel" in Year 3 than in the previous years.

Several factors contributed to this shift. First, the global pandemic shifted the priorities of the community partners from this collaboration to more fundamental community needs, such as getting wireless hotspots to families without internet access. Regular meetings with the Year 3 team took on a higher cost and a lower benefit in this landscape. Second, The Key was largely satisfied with the Year 2 system and had already adapted its workflows accordingly. It was more difficult for them to reimagine workflows when more immediate changes, like bug fixes and feature modifications, would have a greater impact on easing their stress points. Recognizing the need for work on these immediate changes may have contributed to the impression that students were providing "Band-Aid fixes," rather than transforming the project through their own design and implementation contributions. Finally, having a first-time faculty advisor likely played a role—the advisor needed to figure out how to manage the relationship between the students and the community partners, and manage her own relationship and teaching voice with students, while simultaneously managing those relationships in real time. Any faculty advisor will need to manage student–partner relationships differently each year, but more seasoned Comps advisors can fall back on established best practices that they have honed over time via trial and error. In hindsight, the advisor for the first 2 years of the project, herself a seasoned Comps advisor, should have been more proactive in providing more hands-on guidance and onboarding into both project management and Comps mentorship strategies.

As of this writing, none of the Year 3 modifications have been tested or integrated into the production system. The code cannot be tested or rolled out until it is safe for The Key to go back to in-person programming, a date yet to be determined. The Year 3 students have all graduated; even though a couple of Year 3 students agreed to advise the eventual rollout, the testing and rollout will be directed by faculty and students who are not intimately versed in the codebase.

Discussion and Lessons Learned

In this section, we consolidate the key takeaways from the collaboration. Our hope is that these points will prove useful to other institutions considering implementing a similar multiyear collaboration.

Community Partner Impact

The sustained partnership between the students and the community partner yielded both practical and transformative benefits. A long-term partnership allows for a focus on process, instead of only on outcomes. When collaborations happen on short time scales, they need to onboard students

quickly in order to achieve a specific outcome by the end of the course. The burden often falls on the community partner, as the domain expert, to frame this out. A sustained collaboration lent itself to a gradual introduction to the project, with some guidance from the community partner and some observations of "a day in the life" by the students (S. Wopata, personal communication, December 18, 2020). The students did not have to rely on the partner's view and telling, but could integrate their own observations and experiences. Thus, students became equal partners in imagining and planning the eventual solution.

The space to iterate over solutions moves the relationship between the community partner and the students from transactional to transformational. Students, and community partners, gain room to try, fail, reflect on, and retry various approaches, along with room to modify the parameters of the deliverables and the scope of the solution. This method results in less pressure for any deliverable to be "perfect," because both parties know that revisions can occur in the next iteration (S. Wopata, personal communication, December 18, 2020).

This project operated initially under the assumption of data upload and management as the primary bottleneck, and the initial set of solutions concentrated on relieving this bottleneck within The Key's workflow. When the Year 1 students performed a live test of the system, everyone quickly realized that data *entry* posed a bigger bottleneck to the workflow. The partner and students realized that the goal—freeing up staff resources to contribute back to the core mission of serving youth—could not be addressed by simply streamlining data entry; staff mobility when entering data was equally important (S. Wopata, personal communication, December 18, 2020). Rather than losing a year's worth of work and abandoning the effort, the partner recognized that the Year 2 students could build upon these insights and address the new bottleneck. Similarly, once the Year 2 students addressed the data entry bottleneck, the community partner had freedom to envision transformative uses for the data to inform and modify The Key's reach and programming.

Iterative Development

Iterative development is a central tenet of user-centered design, yet the time limitations of a typical term or semester rarely allow students to fully engage in this practice. Effective iterative development reserves time not just for active software development, but also for the necessary space to reflect on project goals and needs, noting how these evolve and change over the lifetime of the project. Removing the time limitations allowed both the students and the community partners to participate in iterative development in ways similar to real-world software development.

The community partners benefited in multiple ways from the iterative development process. The time within each iteration of Comps, and between iterations of Comps, gave the partners space to reflect on their own goals and how these goals were and were not reflected in the current software product. This reflection, along with the need to provide somewhat frequent feedback to students on their design iterations, helped the partners better recognize and articulate their needs—including, and especially, needs that were not apparent at the start of the project (such as the ability to add, modify, and delete activity types). Indeed, the reflection time between the first and second years of the project enabled The Key to recognize the importance of bringing in HCI as a partner on the project—an enabler of systemic change. As we note in the Results section, the need to provide frequent feedback to the student teams imposes costs in time and energy for the community partners—costs that are easier to bear when the community partners have the appropriate bandwidth, and that may change over the lifetime of the project.

The students benefited from participating in a realistic iterative development process that few of our students get to experience in a course. Deliverables like the project proposal became living, breathing documents, rather than academic exercises. Instead of creating requirements from scratch each year, students in Year 2 and Year 3 had the benefit of an existing requirements document and proposal. They used these artifacts to reflect on the choices made by previous groups, match this with their own observations, and *refine* them accordingly. Students brought fresh perspectives to the project that might have been lost the previous year(s) in the scramble to finish deliverables by the project deadline. They had space to notice when project development

deviated from these goals. In addition to learning for their own edification, students simultaneously developed assets to leverage toward a community partner's goals and interests. Finally, from a pure software development standpoint, building upon and maintaining code written by others, for clients, over multiple months requires skills that most of our majors go on to use in their careers.

Students derive numerous benefits from having a project and relationship with the community partner that extend beyond a single class and over multiple years. It allows space for "throwaway" drafts, for learning the hard way, for both sides to envision and reenvision how a tool can best serve a community partner's goals. It more accurately models adult professional life, where failure, and sometimes a series of failures, often leads to innovation.

Project Continuity

Onboarding Students and Teams

Transitioning the project from one Comps team to the next proved surprisingly difficult. Although Comps teams are nominally expected to provide adequate documentation for any code they produce, in reality computer science majors lack the skill to produce documentation that is useful to anyone other than themselves. Even when the faculty advisor primed the students to think about producing a record of development that others could follow, the documentation fell short.

Our solution—designating a mentor from the previous year as the point person for the current year's team—worked most effectively when the designated mentor had a strong grasp of both the codebase and the system architecture. A good choice for this role is the student who served as the technical lead for the project in the previous year.

It is also important for the incoming team to work directly with the codebase right away, rather than reading through the code solely in order to understand it and delaying contributions to it. This philosophy is similar to joining an open source coding project, where new members learn the norms of the community and the codebase by contributing a small code modification, as described in Braught et al. (2018). Future collaborations could follow a similar model.

Similar attention needs to be paid when the faculty advisor changes. We experienced "growing pains" between Year 2 and Year 3, when the switch uncovered the extent to which the original faculty advisor served as "institutional memory" for the project. The outgoing advisor should take an active role in onboarding the new advisor, and should also ensure that advisor-level documentation is clear and complete.

Long-term Maintenance

Long-term software maintenance was a known (and unsolved) issue heading into the project, as it is on many software development collaborations with community partners. We learned the hard way the cost of kicking this problem down the road. We did not have a contingency plan in place for the pandemic-related shutdowns, believing that we would have time the summer following Year 3 to finalize maintenance details. Fortunately, the version delivered by the Year 2 team works sufficiently well for most of the community partner's needs, but in some circumstances not having a working system at the conclusion of the collaboration poses a major issue.

Several maintenance models could work. When the core software is not proprietary, the codebase could be open sourced and community maintained, perhaps with a faculty member or a former project participant as the "point person." Alternatively, student volunteers could maintain and grow the project in a more formal manner, perhaps marshalled by the civic engagement office or as an independent study.

We recommend that groups undertaking a collaboration like ours work out long-term maintenance details up front. They do not need to be 100% complete, and can and should morph as the project proceeds. Having such a structure in place can smooth the eventual code handoff, account for unforeseen circumstances, and provide some measure of guarantee to the partner that they will not be left in the lurch at the project's completion. It is important that the maintenance plan contain information about who is responsible when the software fails or when bugs are discovered, and who bears the cost of factors like website hosting.

Curricular Goals

As a capstone experience, Comps needs to

fulfill a set of curricular goals and requirements for the major. At the end of each project cycle, the faculty advisor weighed the work required to make the software product viable for the community partners against whether this work met the threshold of Comps curricular content. As the required work became less "novel" over the course of the project, these decisions were more murky. It is difficult to determine when a project passes from "active development with curricular benefits" to "maintenance and growth outside the scope of Comps." How to make this distinction remains an unsolved question.

Relationship Building and Maintenance

There are many facets to managing the relationship between the community partner and the student team. Foremost among these is the establishment of trust. The faculty advisor plays an important role in setting expectations—for the community partner and for the students—and in establishing trust with both parties. Meeting with the community partner before the start of the project helps the faculty advisor assess the partner's needs and working style, and sets expectations with the community partner about outcomes, based on the advisor's (likely imperfect) information about individual students' skill sets. Preparing students to meet with the community partner at the project's onset also sets expectations about professionalism, positionality, and so on.

Civic engagement offices also play an essential role. They provide resources to students about the role of civic engagement in their academic exploration, about the community partner relationship, about their positionality, and about many of the other fundamental considerations in critical service-learning (Center for Community and Civic Engagement, n.d.; Mitchell, 2008).

An important aspect of establishing trust between students and the community partner, and in helping students gain a holistic understanding of their work's impact, came from having students perform observations at the community partner site. Being invited into the community partner's space was itself an act of trust on the part of the community partner—trust that students would respect the space and honor the partner's domain knowledge and experience. The observations provided the students with an understanding of place and helped them figure out how the eventual software would fit in with the partner's workflow. Observations also required students to decenter themselves and their expertise, a necessary step for effective and equitable community engagement.

Students need to manage their own relationships with the community partner, including how often to communicate with the partner, how to communicate, and the structure of meetings. Faculty advisors tend to provide "light touch" guidance to the students. Only rarely does the advisor step in with a slightly heavier touch, to assist the flow of initial conversations with the partners or encourage more frequent meetings with the community partner.

Teams tend to have their own "personalities" and ways of working. Such individuality affects not just how well teams work together (Duhigg, 2016; Edmondson, 1999; *Re:Work*, n.d.) but how teams interact with community partners. We saw this play out in both the frequency and the content of team–partner meetings. Year 1 and Year 3 teams met with the partner a couple of times each term, but the Year 2 team met with the partner approximately twice a month. Each team spent time demonstrating the system in its current form and soliciting feedback from the community partner, but only the Year 2 team consistently discussed how features and changes tied back to the community partner's primary goals (rather than just taking the feedback at face value). The team mentor from the previous year can contribute to this aspect of project management by introducing the new team to the cultural expectations and norms set by previous teams. Current teams could then have a framework within which to develop their own working style without jarring the community partner's expectations.

In all 3 years of the project, demoing became a key mechanism of communication between the students and the partner. Demonstrating the current version of the software provided a common language between the students and the partner. Students could translate technical concepts into tangible software interactions, and community partners could communicate technical needs via these same tangible interactions. This highlights a crucial lesson: Differences in specialized understanding are surmountable when students attend to them by facilitating this type of communication.

Managing Expectations

In many cases, the Comps project is students' first experience with independent, client-facing software development. Although many computer science majors complete one or more summer internships in software development, their experiences are likely to be mediated through a manager or mentor. In the Comps project, students interact with the client directly, gaining an entirely different perspective on professionalism and professional software development. Whereas as interns they were likely protected from repercussions of their design and implementation decisions, as Comps students they are fully responsible for all such decisions.

This background, combined with the students' limited exposure to user-centered design and development in our curriculum, skewed students' expectations about the client's interaction with the software. In Year 1, insufficient usability and system testing led to an unstable system, forcing the partner to roll back to the original paper-based attendance system. Students in each year made unrealistic assumptions about how much system troubleshooting clients could and should do. Documentation, both client-facing and developer-facing, improved slightly each year, but was still suboptimal.

Although the multiterm and multiyear nature of the project facilitated iterative development, students did not always take full advantage of this process. Engaging computer science students in best practices in user-facing software development, such as requirements gathering and review and frequent usability testing, is a struggle that was not magically resolved just because students were accountable to real clients. The computer science curriculum, like the curriculum at many higher education institutions, does not focus on nor reward this type of engagement. Computer science majors at Carleton are exposed to this modality in one of the core courses, with the degree of exposure dependent on the individual instructor, and a couple of elective courses.

Curricular changes could address some of these issues, as can targeted mentoring by previous participants and the project advisor. To some extent, however, these are lessons most effectively learned the hard way, in the day-to-day practice of developing software for a client. Those adopting this model should keep this aspect of student development in mind and plan for it when designing and advising such a project.

Final Thoughts

Multiyear, established collaborations between community partners and multiple iterations of the same course provide fertile ground for transformative civic engagement. Long-term collaborations allow for iterative and reflective codevelopment of project goals, artifacts, and deliverables, increasing the potential for transformative impact. They leave space for trust relationships to develop between the partners, faculty advisor, and students, opening up more avenues for authentic engagement. The project described in this article provides a valuable proof-of-concept of this approach. The collaboration demonstrates how thoughtful pedagogy, an active and engaged civic engagement center, and an informed advisor can bring together students and community members to foster real and lasting change in the surrounding community. This project has already had important domino effects. The word about this partnership with The Key has spread, and since, other community organizations have reached out to inquire about computer science Comps groups building systems for them. Building partnerships like the one described demonstrates what's possible and can create ripple opportunities for students as well as organizations. We hope the blueprint we provide here serves as a starting point for similar projects at other institutions, in computer science as well as other disciplines.

About the Authors

Amy Csizmar Dalal *is a professor of computer science at Carleton College.*

Emily Oliver *is the former director of the Media and Movements: Storytelling for Justice program at the Higher Education Consortium for Urban Affairs.*

References

Bopp, C., & Voida, A. (2020). Voices of the social sector: A systematic review of stakeholder voice in HCI research with nonprofit organizations. *ACM Transactions on Computer–Human Interaction, 27*(2). https://doi.org/10.1145/3368368

Borning, A., & Muller, M. (2012). *Next steps for value sensitive design.* In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1125–1134). https://doi.org/10.1145/2207676.2208560

Boyte, H. C. (2009). *Civic agency and the cult of the expert: A study for the Kettering Foundation.* Kettering Foundation.

Braught, G., Maccormick, J., Bowring, J., Burke, Q., Cutler, B., Goldschmidt, D., Krishnamoorthy, M., Turner, W., Huss-Lederman, S., Mackellar, B., & Tucker, A. (2018). A multi-institutional perspective on H/FOSS projects in the computing curriculum. *ACM Transactions on Computing Education, 18*(2). https://doi.org/10.1145/3145476

Center for Community and Civic Engagement, Carleton College. (n.d.). *About us.* https://www.carleton.edu/ccce/about/

Connolly, R. W. (2012). Is there service in computing service learning? In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education* (pp. 337–342). https://doi.org/10.1145/2157136.2157238

Davis, J., & Rebelsky, S. A. (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* (pp. 29–35). https://doi.org/10.1145/3287324.3287508

Dekhane, S., Xu, X., Napier, N., Barakat, R., Gunay, C., & Nagel, K. (2018). Technology focused service-learning course to increase confidence and persistence in computing. *Journal of Computing Sciences in College, 34*(2), 147–153. https://dl.acm.org/doi/10.5555/3282588.3282609

Dell, N., & Kumar, N. (2016). The ins and outs of HCI for development. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (pp. 2220–2232). https://doi.org/10.1145/2858036.2858081

Dombrowski, L., Harmon, E., & Fox, S. (2016). Social justice-oriented interaction design: Outlining key design strategies and commitments. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems* (pp. 656–671). https://doi.org/10.1145/2901790.2901861

Duhigg, C. (2016, February 25). What Google learned from its quest to build the perfect team. *The New York Times*.

Edmondson, A. (1999). Psychological safety and learning behavior in work teams. *Administrative Science Quarterly, 44*(2), 350–383. https://doi.org/10.2307/2666999

Finley, A., & McNair, T. (2013). *Assessing underserved students' engagement in high-impact practices.* Association of American Colleges and Universities. https://secure.aacu.org/imis/ItemDetail?iProductCode=E-HIPSUNST&Category=

Gust, S., & Jordan, C. (2006). *The community impact statement: A tool for creating healthy partnerships.* https://compact.org/resource-posts/the-community-impact-statement-a-tool-for-creating-healthy-partnerships/

Healthy Community Initiative. (2020). *Healthy Community Initiative: Thriving youth. Thriving community.* https://healthycommunityinitiative.org

Johnston-Goodstar, K., & Sethi, J. (2014, July 11). *But do I want to work with you?* [Paper presentation]. "What Went Wrong": Reflecting and Learning from Community-Engaged Research, Minneapolis, MN.

Jordan, C., & Gust, S. (2011). The Phillips Neighborhood Health Housing Collaborative: Forging a path of mutual benefit, social change, and transformation. In L. M. Harter, J. Hamel-Lambert, & J. L. Millesen (Eds.), *Participatory partnerships: For social action and research* (pp. 9–30). Kendall Hunt.

Kuh, G. D. (2008). *High-impact educational practices: What they are, who has access to them, and why they matter*. American Association of Colleges and Universities.

Lee, D. T., Hamedian, E. S., Wolff, G., & Liu, A. (2019). *Causeway: Scaling situated learning with micro-role hierarchies.* In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (pp. 1–12). https://doi.org/10.1145/3290605.3300304

Mitchell, T. D. (2008). Traditional vs. critical service-learning: Engaging the literature to differentiate two models. *Michigan Journal of Community Service Learning, 14*(2), 50–65. http://hdl.handle.net/2027/spo.3239521.0014.205

Parra, E., Haiduc, S., & James, R. (2016). Making a difference: An overview of humanitarian free open source systems. In *Proceedings of the 38th International Conference on Software Engineering Companion* (pp. 731–733). https://doi.org/10.1145/2889160.2892651

Re:Work. (n.d.). Retrieved December 17, 2020, from https://rework.withgoogle.com/print/guides/5721312655835136/

Sabie, S., & Parikh, T. (2019). Cultivating care through ambiguity: Lessons from a service learning course. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (pp. 1–14). https://doi.org/10.1145/3290605.3300507

Saltmarsh, J., Hartley, M., & Clayton, P. (2009). *Democratic engagement white paper.* New England Resource Center for Higher Education.

Strum, S., Eatman, T., Saltmarsh, J., & Bush, A. (2011). *Full participation: Building the architecture for diversity and community engagement in higher education.* Imagining America. https://surface.syr.edu/ia/17

Syeda, U. H., Murali, P., Roe, L., Berkey, B., & Borkin, M. A. (2020). Design study "lite" methodology: Expediting design studies and enabling the synergy of visualization pedagogy and social good. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (pp. 1–13). https://doi.org/10.1145/3313831.3376829

Trebil-Smith, K. (2019). *Perceptions of partnership: A study on nonprofit and higher education collaboration.* Iowa Campus Compact. https://iacampuscompact.org/perceptions-of-partnership/

Vennekens, J. (2020). Service-learning for web technology: Observations from a small case study. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 328–334). https://doi.org/10.1145/3341525.3387414

Voida, A. (2011). Shapeshifters in the voluntary sector: Exploring the human-centered-computing challenges of nonprofit organizations. *Interactions, 18*(6), 27–31. https://doi.org/10.1145/2029976.2029985

Whitney, B., Muse, S., Harrison, B., Edwards, K. E., & Clayton, P. H. (2016). Learning from and with community organizations to navigate the tensions of democratic engagement. *Michigan Journal of Community Service Learning, 23*(1). https://doi.org/10.3998/mjcsloa.3239521.0023.108