Collaboration of Two Service-Learning Courses: Software Development and Technical Communication

Jennifer K. Brown and Joseph T. Chao
Bowling Green State University, Bowling Green, Ohio, USA

jkbrown@bgsu.edu; jchao@bgsu.edu

Abstract

We describe a service-learning collaboration between a software development course and a technical communication course.

Good documentation is essential for easing users' experience with a software system, as well as for introducing maintenance and support personnel to the system. Therefore, one of the main goals of this collaboration was to produce usable documentation to accompany the student-developed software. Additionally, the goals were to provide students with a hands-on learning experience and to provide more time for software development.

We discuss the courses involved, the goals/benefits of the collaboration, the collaboration process, the method of student communication, predicted challenges, assessment criteria, and results and future improvements. Despite the challenges with this first attempt at the collaboration, the results were encouraging and we have gained invaluable insight into how to improve for the future.

Keywords: Pedagogy, agile software development, documentation, active learning, service-learning, collaborative learning, real-world project.

Introduction

Service-learning is an active-learning pedagogy that focuses on community projects that are directly related to students' discipline. As defined in the *Faculty Toolkit for Service-Learning in Higher Education* (Seifer & Connors, 2007), service-learning is a teaching and learning strategy that integrates meaningful community service with instruction and reflection to enrich the learning experience, teach civic responsibility, and strengthen communities.

In a traditional software development course, students work on tightly controlled classroom projects created by instructors. While such tightly controlled projects allow each student the opportunity to practice crucial development skills, real-world projects expose students to situations and learning opportunities that cannot be replicated in classroom projects. Service-learning provides students with hands-on, real-world project experience.

Service-learning as a pedagogical approach to software development courses has been embraced by many (Liu, 2005; Poger & Bailie, 2006; Song, 1996), and

its benefits include not only providing students with real-world experience in their technical and social skills, but also developing in students a sense of responsibility and ownership. Some (Purewal, Bennett, & Maier, 2007; Rosmaita, 2007) suggest that service-learning projects may potentially attract more motivated and higher-achieving students to the computer science discipline, which could be a major benefit for the discipline, especially when computer science student enrollment has been decreasing for the last few years.

Service-learning in software development is not without its challenges, which include, most notably, additional time and organizational demands on instructors, and maintenance needs after project completion. Additional demands on instructors needs to be addressed according to one's circumstance; maintenance could be more complicated. One solution is to have a support center, as suggested by Chase, Oakes, and Ramsey (Chase, Oakes, & Ramsey, 2007). As another solution, this paper suggests that better documentation might ease the burden of maintenance.

Based on the past experience of one of the authors in service-learning software development, user documentation produced by the student software developers was typically low quality and/or scarce. There are two main reasons for not having quality documentation: 1. There is limited time for software development itself, which implies limited time for creating documents, and 2. Students in computer science or software engineering are not trained in technical communication.

In fall semester 2008, the two authors, one an instructor in the Computer Science Program and one an instructor in the Scientific & Technical Communication Program worked together and designed a curriculum that encompassed technical communication in software development. That is, students in a senior technical communication course and students in a software development course worked collaboratively on service-learning software development projects in order to develop software and the accompanying documentation for non-profit organizations.

Our collaboration involved fifty-six students (forty-six from two software development classes and ten from the technical communication class). Students from the software development class were placed into teams and assigned a service-learning project. Then, one or two students from the technical communication class were placed with each team. The teams then collaborated to produce software and end-user documentation for six non-profit organizations.

Courses Involved

As stated above, the collaboration involved two courses: Software Development from the Department of Computer Science, and The Writing Process for Online Documents, a course in the Scientific and Technical Communication Program, which is housed in the Department of English. To better understand the structure of the courses and the collaboration, we have provided a brief description of each course.

CS 464/564: Software Development

Software Development is a project-based software engineering course that aims to provide students with hands-on experience in developing software. Despite the challenges of implementing large projects in a classroom setting, the instructor decided to adopt the service-learning approach in order to offer students real-world project experience.

In the fall semester of 2008, the instructor employed the service-learning model with six real-world projects from area non-profit organizations for students to work on. A total of forty-six students, mostly undergraduate seniors and first-year graduate students, in two class sections were grouped into six teams, one for each project. To ensure the delivery of quality software for the customers, each team was required to follow an iterative and incremental agile software process based on eXtreme Programming (Beck, 2000) and Scrum (Schwaber & Beedle, 2001). This agile

process model was applied successfully by student teams in a similar course taught previously by the same instructor, and was recommended by other educators, such as Alfonso and Botía (2005).

The project was split into five separate time periods (time-boxed iterations) of two to three weeks each. Iteration 0 was designed for customer meetings and preliminary planning, and Iterations 1 through 4 each contained a subset of requirements to be completed, tested, and delivered at the end of the iteration.

All students in a team were to be developers with a shared role in project management; individual software development roles, such as Coach (Team leader), Tracker, End User (student customer), Tester, Designer, Tool Guy, Integrator, Presenter, Documenter, and Installer, were elected within each team in order to instill some personal responsibilities in students, as recommended by Dubinsky and Hazzan (2004). Because this course did not have an associated lab, students scheduled their own meetings and time for project development.

One of the major challenges in teaching this course as a service-learning course is that most of the students in the class have no prior knowledge of software engineering, have never heard of agile software development methods, and are required to complete a large-scale, real-world project with real clients in a short sixteen-week semester. Therefore, the instructor must fast-feed students enough information in software engineering so that they can start the project as soon as possible, which in turn maximizes their time for project development.

ENG 486: The Writing Process for Online Documents

The goal of The Writing Process for Online Documents (Online Docs) is to teach the students how to produce various forms of usable online documentation from beginning to end, through each step in the process: analyzing end users and use contexts, creating a documentation plan and outline, producing the document itself, and designing and performing qualitative usability tests on the document, along with learning to use the necessary technology. The students learn to work with multiple software programs throughout the semester, but they use single-sourcing software to produce their course capstone project, a compiled help file to be placed online.

In the past, the instructor of this course/co-author had not involved her Online Docs students in service-learning or collaboration with another course. Instead, the students produced documentation contextualized in hypothetical scenarios reflective of real-world situations. Or, if the students chose, they could complete a project based on the needs of a client. (The client could be a friend, relative, employer, etc.)

However, that's not to say that service-learning is not an integral and beneficial part of the pedagogy of many technical communication courses. In fact, much has been written about the role of service-learning in the technical communication course: benefits and drawbacks, case studies of projects, and guidelines for implementing a service-learning project.

Moreover, there have been articles published discussing approaches to multidisciplinary client-based projects in the technical communication course, such as we are undertaking (Norman & Frederick, 2000; Tesdell, 2008; Wojahn, Dyke, Riley, Hensel, & Brown, 2001). However, none of these collaborations have specifically involved a technical communication course that focuses on online documentation, and none of the collaborations have been with computer science courses; instead, they have involved collaborations with engineering and art students. They do, however, provide insight into some best practices and anticipated difficulties in a multidisciplinary project involving technical communication classes.

The Collaboration

The entire collaboration was structured around the idea of providing the students from both courses a real-world learning experience. In planning the collaboration, we identified our goals for and anticipated benefits from the project, outlined the process for the collaboration, addressed any predictable challenges or limitations, and established assessment criteria to determine the success of the collaboration.

Goals/Benefits of the Collaboration

We had multiple goals for our collaboration and had anticipated multiple benefits of it as well:

1. Solve the initial problem that instigated the collaboration: poor end-user documentation. We wanted to provide the software users with usable documentation that would enable them to efficiently complete the desired tasks on their new software. The documents created collaboratively included a project plan (revised after each iteration), release notes for each delivery, and a user manual in the form of an online compiled help file.
2. Provide our students with a hands-on learning experience that allowed them the opportunity to manipulate the tools they would be using in their careers and to learn to navigate the production processes in their respective fields.
3. Provide our students with a collaboration/communication experience similar to what they will have in their careers: working with clients and client needs, collaborating with peers, and navigating communication channels.

   Most software developers and technical communicators do not work in isolation, but must collaborate with multiple contributors and stakeholders: they must gather information from their co-workers, supervisors, and outside clients; they must keep everyone involved updated on the status of the project; they must learn to coordinate schedules; they have to help with distributing responsibility for a project; they have to solicit feedback from those involved to determine the success of the project, and may even have to develop new ways to solicit that feedback; and they simply have to be "good team players" throughout their careers. This project provided them with an opportunity to exercise those skills in a real-world project.

   An additional element of the collaboration/communication experience is the opportunity to work on a multidisciplinary team and, as part of that, figuring out how students of the other discipline fit into the project and what strengths they bring to the collaboration.
4. Secure more time for software development. A sixteen-week semester offers little time for software development students to plan the software development, develop the software, test it, and document it. Moreover, in the one author's experience, many computer science students simply do not like writing. In a survey conducted on the first day of the computer science class, most of the students (thirty-seven out of forty-six) reported they believed that they are good writers; however, only twenty-six students stated that they enjoyed writing.

   On the other hand, those students in the Online Docs course enjoy writing and had taken prerequisite classes to develop the fundamental skills of technical communication. Therefore, the collaboration would reduce the burden on the computer science students to produce end-user documentation, when, most likely, they will not be responsible for end-user documentation in their careers. Instead, the computer science students communicated the concepts to the technical communication students, both verbally and through programmer notes, who then, based on user assessments and technical communication background, produced the end-user documentation.

To be certain, this process does not completely alleviate the computer science students of written communication responsibilities, but instead requires them to use written communication in ways that are more reflective of the ways they will be required to communicate in their future careers. This process also allows the computer science students more time for project development.

5. Enhance the usability of the software. Technical writers often add an extra layer of usability testing to software development (Fischer, 1999). As the writers experiment with the software in order to understand it and then be able to document it, they can find errors or opportunities for improvement before formal quality assurance testing.

   Additionally, if the technical writer simply does not understand an aspect of the software, it's possible the end-user will not understand it as well. In that situation, the writer can serve a as a liaison for the end-user. The writer and developer can solicit input from the end-user to determine if a modification is needed; or, the writer and developer can work together to determine whether or not to modify the software and then wait until the qualitative usability test to gather end-user input on the issue.

The Process for Collaboration

Due to the strict time constraint in a semester, the first thing the students learned about their projects was the deadline, which is the same situation as in the real world. Each project contained five iterations, and the deadline for each iteration was determined by the instructor. The students were responsible for deciding the extensiveness of the product, based on user needs and time constraints. Because the students were using the agile approach to software development, they determined what features and functions would be included in each iteration and delivered to the client.

Students in both the software development course and the technical communication course were introduced to the collaboration concept well in advance of taking part in it. Once the computer science students defined their group's project, each group was paired with one or two technical communication students, depending on the needs of the project.

Microsoft Visual Studio Team Systems was used for project collaboration and to allow the Online Docs students to access the source code in order to document the functionality as it was being built.

Scope/nature of projects

The Software Development students worked with the clients in order to determine the scope of each project. The scope was determined based on the needs of the client, the size of the development team, and the final completion deadline. We found that the scope of the projects varied somewhat significantly, depending on students' confidence with their ability to develop software from conception to completion in 12 weeks (the first 4 weeks of the semester are spent learning the building blocks of software development in order to begin the project).

Below are the types of software systems developed by the teams:

- Tracking system to record and monitor domestic-abuse cases
- Employee database system
- System for reporting services provided to clients at a women's shelter
- Information system for a university service-learning office
- System to match compatible students as partners for extracurricular activities at a middle school
- E-voting system to be used by judges of high school speech competitions

The technical communication students developed two different types of documentation: final end-user documentation and release notes. The technical communication students were also involved with editing the project plan.

The final documentation was required to contain an overall introduction to the software system, contextual introductions to each task (i.e., an explanation of why the user would perform a given task and what the result should be), step-by-step instructions with accompanying screenshots, and troubleshooting information.

The final documentation was published as a compiled help file chunked by user tasks, which makes it difficult to assess the scope (in terms of length) of the documentation. However, when the compiled help files were formatted and published as printable Word documents, they were approximately 8–10 pages in length.

The release notes were updated with each iteration, so that the release notes were built incrementally, and eventually contained most all of the information that would then be translated into the compiled help file.

For the project plan, the technical communication students' role was simply to edit the plan for usability so that the client could understand all of the concepts within it.

Schedule

The due dates for the software and documentation were developed around the constraints of a sixteen-week semester, which limited the time to work on each iteration to no more than two weeks. The final iteration was due at the beginning of the last week of the semester.

Method of student communication

As the teams worked on their projects, several communication and collaboration tools, such as email; wikis; a file exchange server; and Visual Studio Team Systems, which includes a versioning tool, were available for them to communicate with one another and document the work each had done. Additionally, the instructors were able to use this documentation to help evaluate each student's contribution to the project.

Predicted Challenges/Limitations

To better plan for the collaboration, we attempted to predict any challenges or limitations of the project. Based on past experience, we forecasted that time constraints would be the greatest challenge to the project: our students usually have full schedules and to coordinate group meetings that all group members could attend would be difficult.

To do as much as we could to address this anticipated challenge, we designed our courses so that the collaborative service-learning project was the course focus and all assignments were building blocks to complete the project. That way, each class assignment put the students closer to completing their project. Moreover, students were not doing extraneous work and could therefore have more time available to meet outside of class. We also designated some days for our students to work during class time on their projects.

Assessment Criteria

To judge the success of the project and to help determine how to improve it in potential future collaborations, we established a set of assessment criteria:

1. Quality of documentation: Evaluated against the guidelines for usable, quality end-user documentation, was the documentation developed by the technical communicators better than the documentation produced by the Software Development students from previous semesters?
2. More usability issues addressed earlier in development: Did the technical writers help catch usability issues that otherwise would have gone unnoticed until a later time when a change in development would be more detrimental and time-consuming?
3. Student feedback: Did the students agree that they learned something valuable from the project itself and the collaboration? We used an anonymous survey at the end of the semester to help assess this aspect.

Results and Future Improvements

We found that the collaboration offered many surprises that offer new insight for future attempts at this collaboration. We will address the results in relation to the criteria that we had previously established.

1. Quality of documentation. Assessing the produced documentation through something of a "technical-communication heuristic evaluation," including task-oriented writing, contextual introductions, results of actions, etc., as well as basic elements such as grammar, usage, and syntax, the documentation produced by the technical communication students was indeed better written than the documentation of the software developers in previous semesters.
2. More usability issues addressed earlier in development. We found that the technical communication students, not completely convinced of their credibility in playing the role of "user as expert," were hesitant to present their "subjective" usability concerns to the developers. They did, however, help enhance system usability through objective means, such as completing validity testing, or by discovering that a system did not work in internet browsers other than Microsoft Internet Explorer.
3. Student feedback. It is clear that the software development students valued the service-learning project as a means of providing them an experience similar to what they will undertake in their careers. In the end-of-semester survey, 100% of the 46 students agreed that they had learned skills in the class that were applicable to the real world; more than 87% agreed with the statement that their participation in the course would improve their chances of landing a job after graduation.

   Moreover, in their responses to the open-ended questions of the survey, they expressed a sense of fulfillment and pride in completing their projects because the systems they created would actually be used. Additionally, they expressed a feeling of worth and satisfaction in being able to contribute to a non-profit organization to help better a community.

   In questions regarding the collaboration with the technical communication students, only 25 out of 46 software development students agreed (17 were neutral) that "Our collaboration with the technical communicators on our team has been effective, purposeful, and useful." However, 32 out of 45 students agreed (11 neutral) with "Collaboration with the technical communicators was a good idea and should be continued for this course in the future."

   The technical communication students were less likely to say that they found the overall project useful to them. For the end-of-semester survey questions listed below, only five of the seven students who completed the survey agreed with the statements:

   - I believe this collaboration has given me an experience similar to collaborating in a work setting.
   - I believe that what I have learned from this collaboration will be useful in my future career.

However, six of the seven students agreed with the statement "I have gained beneficial skills in working with a team through this collaboration."

These responses could be due to a number of factors, but the technical communication students unanimously agreed on four aspects, indicating these aspects could be major contributors to their disappointment with the project: (1) They were uncertain of their roles and responsibilities, and the understanding they did have often times was not the same understanding that the software development students had of the technical communication students' roles and responsibilities. (2) They often did not feel they were an integral part of the team. (3) The software development students often completed their portion of work the night before it was due, leaving the technical communication students to stay up all night to complete the documentation. (4) Having the Online Docs class delivered completely online was not the ideal delivery method for working with this collaboration. (The delivery method was established prior to the conception of the collaboration.)

In order to address these issues in future collaborations, the instructors will do the following:

- Visit each other's classes to present how their own students will be interacting with the teams and what role they will play.
- Allow the software development students a portion of class time to work with their team members, and assign the technical communication students into teams partially according to their schedules to ensure that at least one of them from each group can attend the teamwork time.
- Provide each group with a list of discussion points at the beginning of their collaboration. One of those points will be on the students' roles to ensure that both groups of students within the team have the same understanding of everyone's roles.
- Require a two-day window between development completion deadline and the iteration due date with accompanying documentation.
- Incorporate the technical communication students into the initial client meeting so they have a better big-picture understanding of the goals of their team's project and so they feel more comfortable providing input, based upon that better understanding.
- Deliver the Online Docs class as either a face-to-face or hybrid (half online and half face-to-face) course rather than online.

Conclusion

This paper has described a collaborative service-learning project between a software development class and a technical communication class, defining the rationale behind the collaboration, the goals and anticipated benefits, the process for the collaboration, the assessment criteria to determine the success of the collaboration, and the final results. The collaboration was initiated to solve the problem of poor end-user documentation developed by Software Development students to accompany their service-learning software projects. We found that, through this collaboration, (1) the documentation did improve, according to a heuristic evaluation of end-user documentation, and (2) the students, overall, found value in the collaboration. Therefore, we will continue the collaboration, improving with every attempt, learning from the challenges of this first endeavor.

References

Alfonso, M. I., & Botía, A. (2005). An iterative and agile process model for teaching software engineering. *Proceedings of the 18th Conference on Software Engineering Education and Training (CSEET'05), Ottawa, Canada*, April 18-20, 9-16.

Beck, K. (2000). *Extreme programming explained: Embrace change*. Addison-Wesley.

Chase, J. D., Oakes, E., & Ramsey, S. (2007). Using live projects without pain: The development of the small project support center at Radford University. *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education Conference. SIGCSE'07, Covington, Kentucky*.

Dubinsky, Y., & Hazzan, O. (2004). Roles in agile software development teams. *XP/Agile Universe 2004, LNCS 3134*, 32-42.

Liu, C. (2005). Enriching software engineering courses with service-learning projects and the open-source approach. *The 27th International Conference on Software Engineering (ICSE'05), St. Louis, Missouri,* May 15-21.

Norman, R., & Frederick, R. A. (2000). Integrating technical editing students into a multidisciplinary engineering project. *Technical Communication Quarterly, 9*(2), 163–189.

Poger, S., & Bailie, F. (2006). Student perspectives on a real world project. *Journal of Computing in Small Colleges, 21*(6), 69-75.

Purewal, T. S., Bennett, C., & Maier, F. (2007). Embracing the social relevance: computing, ethics and the community. *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education Conference. SIGCSE'07, Covington, Kentucky.*

Rosmaita, B. J. (2007). Making service learning accessible to computer scientists. *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education Conference. SIGCSE'07, Covington, Kentucky*.

Schwaber, K., & Beedle, M. (2001). *Agile project management with Scrum*. Prentice Hall.

Seifer, S. D., & Connors, K. (Eds.). (2007). *Community campus partnerships for health. Faculty toolkit for service-learning in higher education.* Scotts Valley, CA: National Service-Learning Clearing House.

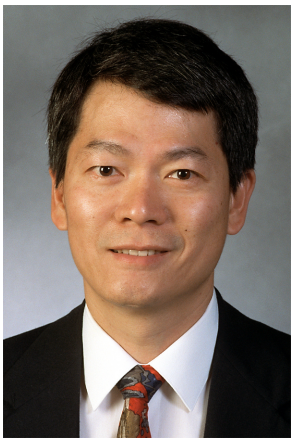
Song, K. (1996). Teaching software engineering through reallife projects to bridge school and industry. *SIGCSE Bulletin. 28*(4), 59-64.

Tesdell, L. S. (2008). Technical communication, art, and printing services: What are the pedagogical connections? *Business Communication Quarterly, 71*(2), 216–221.

Wojahn, P., Dyke, J., Riley, L. A. Hensel, E., & Brown, S. C. (2001). Blurring boundaries between technical communication and engineering: challenges of a multidisciplinary, client-based pedagogy. *Technical Communication Quarterly, 10*(2), 129–148.

Biographies

Jennifer Brown is an instructor in the Scientific & Technical Communication Program within the Department of English at Bowling Green State University (BGSU). Jennifer teaches courses in introductory technical communication, online documentation, and professional editing. She has been teaching at BGSU since 2005, and previously worked as a technical writer/trainer for a software company. Jennifer has also freelanced as a technical writer, editor, instructional designer, and consultant for individuals and corporations. She earned her MA in Technical Communication from Minnesota State University, Mankato.



Dr. **Joseph T. Chao** is an Associate Professor in the Department of Computer Science at Bowling Green State University. He has taught courses in all aspects of the software development lifecycle, including programming, systems analysis and design, database systems, usability engineering, and software engineering. Prior to entering academia, Dr. Chao has more than seven years of industry experience in software development, including three years as Director of Software Development. His research focus is on software engineering with special interests in agile methods, programming languages, and object-oriented analysis and design. Dr. Chao is the Director of the Agile Software Factory at Bowling Green State University, which he founded in 2008 with a grant from the Agile Alliance. The Factory provides students with service-learning opportunities in software engineering. Dr. Chao holds an M.S. in Operations Research from Case Western Reserve University and a Ph.D. in Industrial and Systems Engineering from The Ohio State University.