

Lessons Learned From Different Types of Projects in Software Engineering

Jennifer A. Polack-Wahl
University of Mary Washington
1301 College Ave
Fredericksburg, VA 22401-5300
(540)-654-1318
polack@umw.edu

ABSTRACT

Educators teaching software engineering face a large problem when trying to assign “real world” projects. Should the instructors make up “real world” projects or have industry propose projects? I have tried both of these, with variable success and failure. Instructor-designed projects are great from an academic standpoint but students show little interest in them because “real people” will not use the final product. Working with industry gives the student a real product to develop but can deteriorate academically. Projects solicited from not-for-profit organizations and university departments have succeeded in our course further than either of the aforementioned project sources. Students worry less about the grading criteria and more about what they are learning and how to use it to improve their project design and implementation. This paper shares experiences with different types of software engineering projects and shows the positive affects of projects originating from not-for-profit organizations and university departments.

Categories and Subject Descriptors

K.3.2 [Computer Education]: Computer Science Education

Keywords

Software Engineering, Projects

1. Introduction

The Software Engineering Course was based on creating several virtual software companies that were composed of teams and team leaders working on “real” world projects [4]. The course was designed so that assignments allowed the students to experience project situations that had relevance to the real world [7]. The “clients” told the student developers what the system was to do, who would use it, and generally, how it should work. (The terms “real world” and “clients” are defined in each section corresponding to the three types of projects discussed in this paper.) Throughout the semester, students developed their software following given guidelines. Each group met with their clients to discuss any misconceptions and clarifications, report progress, and show any demos that were necessary. There were several milestones throughout the semester. Students had to turn in

reports, make presentations, and communicate with each other.

The academic objectives of the projects are for students to learn the basic software engineering principles: requirements gathering, development of specifications, analysis of the problem, designing a solution, and implementation of the solution. Testing and maintenance are objectives of the course but due to time constraints they rarely are applied to the project. The business environment objectives include working and interacting with clients, and team management experience. The prerequisites to the course include CS I and II, Advanced Data Structures, Discrete Math and Computer Systems and Architecture.

The students who participated in the course were mostly juniors but included some second semester sophomores who received AP computer science and math credits. All students majored in computer science and several were double majors in business, mathematics, and economics.

2. Instructor-designed projects

As educators can we really re-create the uncertainty and mishaps that occur in a real world project? Can we create a project that is real world but do-able in a semester? The answer to these questions is maybe; is that good enough? This section attempts to answer these questions by describing experiences with instructor-designed projects.

Instructor-designed projects usually fall into three categories: databases, point of sales, and games. These projects were extremely successful in the area of design and implementation, but unsuccessful in reproduction of the uncertainty and continual change present in any “industry” project [3]. Some project examples are shown in Table 1.

Table 1 Example of Instructor Design Projects

Point of Sales	Databases	Games
Book Store	Alumni relations	Battleship
Antique Store	Volunteer Tracker	Concentration
Framing Company	Employee Payroll	Yahtzee

The students were given a brief description of each project at the beginning of the semester. Upon project selection, the development team interviewed the client for more information. The role of the “client” has been played by either one of the other student development teams or the instructor.

When students served as clients, the development team progressed through the software engineering process without most of the typical software engineering problems, such as changing requirements or miscommunication between clients and developers. The student clients minimized problems out of loyalty to fellow students and usually agreed with the developer on all decisions. In an attempt to make the student clients act more realistically, they were graded for their role as the client. First peer review was used to grade the student client, which led to the same loyalty issue and therefore proved ineffective. Next, faculty observation was used and this methodology improved the client/developer relationship but was highly time-consuming for the instructor.

One example of instructor-designed projects using student clients and peer review was the game Concentration. The clients were told who they were pretending to be prior to any client/developer meetings: a daycare provider who wanted the children to improve their short term memory and skills in matching like objects. Seen from the development side, the project was successful, as the development team produced a working game that had multiple levels and scoring capabilities. But the project was unsuccessful overall, because the clients never questioned the developers’ choice of images used for matching. The images used were artistic but not relevant to children in daycare. When questioned by the instructor, the clients responded that the images were created by one of the developers and they wanted to support her artistic talent.

When the instructor played the role of the client, the process improved significantly. It was easy for the client to change requirements in the middle of the process or act as if they knew nothing about computers, but problems still existed. One problem occurred when students would ask questions. The client would be faced with an identity crisis: are they

the client or the instructor? Students looking for academic guidance who got a client answer were visibly frustrated, but a client who consistently gave academic guidance lessened the potential for the software engineering mistakes that occur during a real world industry project. The second problem was that the instructor had to double their workload, grading and reviewing all work as the instructor as well as dealing with meetings, reviews, questions, and comments as the client.

If there is no viable alternative, this methodology has proven to be one of the best implementations, and the students get a “pseudo real world” feeling. Two problems that speak against using this methodology are that the professor does not fully model the frustrating process that occurs when dealing with real clients, and the professor’s workload increases significantly [5]. Perhaps an even more important deterrent for instructor-designed projects is that students exhibit a lack of interest in these projects because they lack applicability outside of the classroom [2]. The students care more about how the instructor is going to grade them than about truly learning how to build software. Therefore, my contention is that students need to work on a real project in order to gain a true learning experience about software engineering prior to graduation.

3. Industry and government sponsored projects

Computer science professionals will always say that it is beneficial for students to work on solutions to meaningful problems, meaning software that can be used by real clients [5]. This section of the paper discusses projects that have been sponsored by real world companies and/or governmental agencies. Companies are defined as any organization providing goods and/or services involving financial profit. Using local contacts, the software engineering course has recruited many businesses to sponsor a variety of projects. Some examples are shown in **Error! Reference source not found.**

Table 2 Industry and Government Sponsored Projects

Business Type	Size	Project
National Capital Management Corporation	Large	Web-based Building Outlay
Local Police Department	Medium	Student Parking Ticket Tracking
Family-Owned Deli	Small	Sandwich Ordering

The main problem was finding external clients who have suitable projects for all student groups. Students without external clients often felt that they were getting the short end of the stick. Either they were not getting a resume booster, or their software was never used in the real world. The only exception to the second complaint was teams that programmed games. These development teams usually self-distributed the game and then regularly witnessed other students playing the game in the computer labs.

The first time external projects were used students had minimal success; out of seven projects only one was deemed successful (the software was used for six months after delivery). One reason for failure was the preliminary project selection process. Initially, project proposals were accepted up until the day of classes. The clients agreed to fill out two group evaluation forms and meet with the student teams at least four times over the semester. A one-page project description of accepted proposals was given directly to all students by the instructor prior to the students' selection of a project. The students did learn the software engineering process but were dragged into complicated industry politics and/or told software developing techniques contradictory to those discussed in class. In fact, many of the larger corporations sponsoring projects encouraged a build-and-fix model even though this was specifically prohibited in the initial participation agreement.

Currently all project proposals are reviewed and accepted or rejected six weeks prior to the beginning of the course. During those six weeks the instructor and clients communicate regularly to fine-tune the project requirements. The instructor does an initial informal requirement specification and reviews this with the client. Any major wrinkles are worked out and the result is a one-page project description that is distributed to the course participants prior to project selection. Furthermore, the client signs a confidentiality, non-disclosure, and commitment form [1]. This process has improved our success rate with external clients because it has improved client/instructor communication, client expectations, project feasibility, and student/client communication. But it has not significantly changed the successful completion of a software product that is used after delivery.

Out of the three projects given as examples in this section, only one project was deemed successful both in client expectations and actual software implementation, the sandwich-ordering system. The

reasons for success in this project and similar successful projects can be categorized as followed:

1. Well-defined projects. Requirements usually changed, but almost all changes did not affect the critical path. The actual methodology of how to complete a critical path requirement had changed, but not the actual requirement.
2. One member of the project team interacted with the client on a weekly basis and offered to train or fix problems after the course was over.
3. Students primarily managed projects internally, students worked out group issues instead of the instructor intervening.

In contrast to the sandwich-ordering software, the Capital Outlay Project was a complete failure. Out of the four participants working on the project, two passed with reasonable grades, one passed with an extremely low grade and one student failed. The final system designed by the students was completed without the client's participation. This project and similar projects that were deemed failures had the following attributes:

1. Requirements had major changes and/or became too large for a one-semester project.
2. Teams had severe problems with members behaving as a team [6]. The major problem was communication or the lack of communication. Some students become dictators while others in the group become slackers. In all cases when the group has had a slacker, that person either received a 'D' or 'F' in the course. This is a problem that can affect any type of project and is not specific to industry-sponsored projects.
3. Clients' requests became outrageous and/or the client tried to take over the teaching process. In many of these cases, either the instructor decided that it was best to sever ties or the clients felt the students took too much of their time and decided to end their commitment. In both cases, the instructor took over as the pseudo client.

Sometimes projects are neither successes nor failures but instead fall somewhere in between. Such a project was the Student Parking Ticket Tracking System. The project was set up as a mechanism for the local police department to receive payment for student parking tickets before students were allowed to register for classes the next semester. Our contact was with the local police department, which had already gotten an

agreement from the university that both parties would share the necessary information, in order to support the project. Breakdown in communication between the university and police department made it impossible to implement a system with real data and the dual environment. The development team implemented the police portion, to search for and retrieve college residents who received tickets, but were never given access to the data needed to implement the college portion of the project. So, part of the system was successfully developed and could have been used if the police department had been able to make use of the data produced by the software, but unfortunately local politics between the police department and the university made this impossible. Other projects that involved more than one external client had similar results: parts of the project were successful but had no meaning without the other portions of the software.

Overall, I found most successful those projects that had one contact person with authority, a client who understood the academic environment, and who genuinely proposed a project to open lines of communication between the college and themselves. In summary, seven external clients out of the last ten clients later confessed they expected more than they received, even though they were told upfront that only 15% of previous external projects had been completed. Some of the reasons for disappointment are expressed in the following comments:

1. The project was small and the client could have done it themselves in a month, if they had the time.
2. The client had seen other software that was similar to their idea; therefore, it must be easy.
3. The client had done something similar while they were in college; therefore, it must be feasible.

Many educators would argue that carefully chosen and well-supervised industry projects can work and have worked. While this statement is true, many universities may find it hard to launch a working relationship with industry, especially when the course is only looking for only a semester-long project. In addition, "industry partnerships do not offer the potential for students to get personally committed to developing the best possible solution within the time available" [2]. Other options still exist and these options have proven successful not only from a teaching perspective but also from the students' perspective.

4. Not-for-profit organizations and university departments

In the last two years, the course has had enormous success with projects from not-for-profit organizations and university departments. Some examples are shown in Table 1.

Table 1 Project from Not-for-Profit and University Departments

Organization	Project
YMCA	Nursery Tracking
Economics Department	Alumni Database
Education Department	Student Teaching Placement
Rappahannock Area Office on Youth	Participants and Programs Database
Fredericksburg Food Bank	Volunteer Tracking

The benefits of a project from a not-for-profit organization can be categorized as followed:

1. They usually do not have money to spend on software or upgrades, and therefore are extremely grateful for the possibility of free software.
2. They are willing to make a weekly meeting commitment with the development team.
3. They understand that they may or may not get a working product.
4. They are used to working with volunteers.

The benefits of projects from a university department can be categorized as followed:

1. The students have access to the same network, tools, and programs as the sponsoring department.
2. The department client is already use to working with student projects.
3. The students have easy access to the client.

Over the last couple of years, the course has had 11 successful projects out of 15. Success is defined as the software being used for a minimum of one year after delivery. Two of the organizations, the Fredericksburg Food Bank and the Rappahannock Area Office on Youth, have worked with the software engineering course on more than one occasion. One sponsor said, "Our office has had a great experience with your students and is extremely appreciative of their and your assistance." Another sponsor wrote, "The staff at the food bank is eager to participate again. . . . The

software has made a difference in the food bank's ability to serve the needy of the community." Students who participated in these projects received community service recognition and excellent letters of recommendation upon graduation.

Out of the four unsuccessful projects, three groups could not work as a team and the other project produced a working system, but it was missing many major requirements, therefore making the software incomplete. The incomplete software was developed for the University of Mary Washington's Education Department. The software was meant to aide the department in matching student teachers with different student teaching assignments. The system was used for multiple semesters to match many student teachers to a particular school and class. Any refinements were done by hand after the system produced its initial matching. The education department stated, "We were extremely pleased with the software that was developed, even though it wasn't complete. . . . We use to match 75 student teachers, with different needs, to 225 classes, all by hand. The software had cut the time it took by ¾." These students continued to work on the project as community service, but also because they were hooked. The incentive for them to continue working on the project was stated as "a great real world experience . . . and we have easy access to the client."

Even though the course has found a gold mine in these types of projects, we still experience problems. Most of our problems occur when soliciting organizations/departments for project proposals. Many times the person submitting the proposal does not understand what is meant by "software." Many submissions are simplistic introductory programming problems or simply the creation of web pages. To try and minimize the wrong kind of submissions, I have sent out emails explaining what is considered a good versus a poor project. Additionally, I have given examples of past valid and invalid project submissions. Unfortunately, this has not decreased the number of poor project proposals; consequently, the instructor usually has twenty or so submissions to weed out.

5. Outcomes

In order to gain some perspective about the effects of project type on graduates, I conducted a survey about how the software engineering course played and plays a role in their job search and current jobs. Forty-two graduates responded: 17% of the respondents worked on a not-for-profit project, 24% worked on instructor-designed projects, 24% worked on on-campus department projects, and 36% worked on industry

projects. All not-for-profit and on-campus projects were successfully implemented. Seventy percent of the instructor-based projects were successfully implemented. Finally, 30% of the industry projects were successfully implemented.

One question asked was what component of the computer science program was a key factor to their first post graduate job. The number one answer (67% of all those responding) was their software engineering portfolio; figure 2 breaks the results down by project type. The students are divided according to what type of project they worked on in the course. Students who worked on our "gold mine" projects (not-for-profit and university departments) really believed that their projects were one of the key components to getting hired. While the number for instructor-designed projects is high (60%), many of these students stated that they were unprepared for real world client communication and were shocked by the amount of changes and miscommunications that occur during a project lifecycle. Finally, the majority of students who had worked with real world companies and governmental agencies stated that they had "nothing to rave about [when it came to their projects]" therefore they only talked generally about what they learned in the software engineering course.

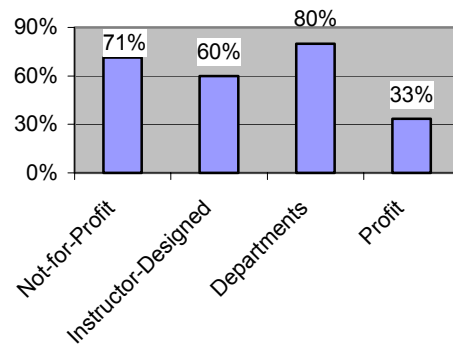


Figure 1 Percentage of students who attributed their software engineering project portfolio as the key to their first job.

Of all the students surveyed, 79% said they use a large majority of the skills learned in the software engineering course. Out of that 79%, the majority of students said they use in their current careers the methods they learned for analysis and design, as well as the lessons they learned from working in a team.

6. Conclusion

This paper outlines past experience with successful and unsuccessful projects, and proposes the use of projects from not-for-profit organizations and university departments for better success. Developing a software engineering course that provides students with a realistic software development experience can be difficult. Finding the right real world projects is not an easy task, but I believe that the lessons learned from my course show that not-for-profit and university department projects make it easier to learn software engineering methodology and the business skills that goes along with software engineering development. Students who have taken the course using these types of projects have not only learned the material successfully, but in addition have been motivated to create a product that would not only receive a passing grade but would also be useful to their clients; and have in the end felt appreciated for their hard work.

7. References

- [1] Bair, Borstler, Lethbridge, and Surendan. Client Sponsored Projects in Software Engineering Courses. In the Proceedings of the Thirty-Fourth SIGCSE Technical Symposium on Computer Science. ACM 2003, pp. 401-402.
- [2] Buckley, Kershner, Schindler, Alphonse, Braswel. Benefits of Using Socially-Relevant Projects in Computer Science and Engineering Education. In the Proceedings of the Thirty-Fifth SIGCSE Technical Symposium on Computer Science Education. ACM 2004, pp. 482-486.
- [3] Chamillard and Brau. The Software Engineering Capstone: Structure and Tradeoffs. In the Proceedings of the Thirty-Third SIGCSE Technical Symposium on Computer Science. ACM 2002, pp.227-231.
- [4] Concepcion. Using an Object Oriented Software Life-Cycle: Model in the Software Engineering Course. In the Proceedings of the Twenty-Ninth SIGCSE Technical Symposium on Computer Science. ACM 1998, pp. 30 - 34.
- [5] Gabbert and Treu, "Reality Check: Working with Meaningful Projects in and out of the Classroom." The Journal of Computing Sciences in Colleges CCSC. 2001 Volume 17, Issue 2, pp. 191-198.
- [6] Stein, "Using Large versus Small Group Projects in Capstone and Software Engineering Courses." The Journal of Computing Sciences in Colleges (CCSC). 2002 Volume 17, Issue 4, pp. 1-6.
- [7] Villarreal and Butler. Giving Computer Science Students a Real World Experience. . In the Proceedings of the Twenty-Ninth SIGCSE Technical Symposium on Computer Science. ACM 1998, pp. 30 – 34