Grabowski, L.M.; Reilly, C.F.; Lawrence-Fowler, W.A., "Emulating a corporate software development environment through collaboration between student projects in six courses," *Frontiers in Education Conference (FIE), 2014 IEEE* , vol., no., pp.1,7, 22-25 Oct. 2014

DOI: 10.1109/FIE.2014.7044137

The published version of this paper is available from IEEE Explore:
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7044137&isnumber=7043978

© 2014 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Emulating a Corporate Software Development Environment Through Collaboration Between Student Projects in Six Courses

Laura M. Grabowski*, Christine F. Reilly, and Wendy A. Lawrence-Fowler
Department of Computer Science
The University of Texas – Pan American
1201 West University Drive
Edinburg, Texas 78539
USA
Email: grabowskilm@utpa.edu*, reillycf@utpa.edu, wfowler@utpa.edu
*Contact Author

Abstract—**Corporate software development often takes place within a complex organizational structure, potentially encompassing many individuals. With constant improvements in network and communication technologies, those organizations may be widely distributed through time and space. In computer science and education, group projects are typically included as part of an undergraduate and graduate engineering curriculum to help prepare students for the dynamics of the business workplace. However, the groups tend to be much smaller than those typically found in the international corporate world where engineers are required to participate in large groups that are dispersed through geography and time zones. We describe a collaboration between student projects in six courses that aims to emulate such an international corporate software development environment. The collaboration brought together three faculty members and over 90 undergraduate and graduate students to work on a software project for a real client. Through this experience, we learned valuable lessons regarding the importance of communication and coordination between the faculty and student participants in a large–scale project.**

I. Introduction

Complex organizational structures and collaboration are commonplace in the corporate world of software development. The internet facilitates the dispersion of projects across time and space; the globalization of software development and multi-site development introduces additional challenges. This leaves educators with an even greater challenge in determining how best to prepare students for a work place that demands communication, collaboration, cooperation, and coordination among many individuals whose work is heavily dependent on the work of others. We must consider how best to develop skills to address tensions between group and individual interests, the complexity and distribution of responsibility, the potential for competition to overshadow collaboration, or the possibility of teams to go off task or develop a skewed idea of the project as a whole. Our goal is to prepare our students to become managers that are able to choose good guiding principles and thwart the tendency of their teams to make premature decisions.

Across the board in academia, team oriented projects are used to prepare students with the skills that they need to be successful in the 21st century. To capture the context of current practice in industry, a number of courses have incorporated projects that emulate real-world experiences, sometimes involving real clients [1]–[11]. There is a large body of literature that shows that this approach is successful in helping students develop skills including communications, problem-solving, planning, organization, initiative, and self-management and suggests that the courses are highly effective in preparing students for the corporate work environment as well as consolidating learning [12]. The introduction of formal software engineering knowledge together with concurrent technique and skill development, can bring valuable experience to both students and faculty [3]. While it is critical that software engineering students leave the university with the technical and soft skills expected by employers [13], a single semester or two of course work may not prepare them to move into the workplace as a software engineer.

As defined through projects such as the Software Engineering Body of Knowledge (SWEBOK) [14] and SE2004 Software Engineering Education Knowledge (SEEK) [15], courses in software engineering should cultivate a broad range of capabilities from problem solving and project management to professional practice and ethics. Even with this guidance for the design of software engineering curriculum, educators are often left to introduce the breadth of the enduring principles of software engineering in one and only one course [16], [17]. Even so, through continual feedback and refinement of the courses, faculty move toward the goal of successfully transforming their students from assignment mode workers to individuals who can work in a team and sustain performance over an extended period of time.

In the spirit of continual process improvement, we incorporated the idea of a multi-site software development project into the existing project-based curriculum. Working with a real client, students were placed into a software development environment which required them to work collaboratively over an extended period of time on a concrete problem where the work was distributed across multiple teams, separated by time and space. Students had to shift their focus from single assignment mode to a focus that allowed them to develop some understanding of all elements of the problem and problem solution. To be successful, students had to make use of existing knowledge and acquire new knowledge as needed. They had to shift their approach from a simple code and deliver mode to one that required an understanding of process and fostered interpersonal skills.

In this paper we describe the organization and implementation of this project, followed by the challenges we faced and the lessons we learned. Our experience is likely to be useful to faculty who are seeking to provide their students with an engineering design and development experience that mimics the complex organizational structures common in the corporate world.

II. Educational and Program Context

The project took place at the University of Texas-Pan American (UTPA), a primarily undergraduate Hispanic-serving university in Edinburg, Texas, located in the lower Rio Grande Valley of Texas. UTPA serves an area on both sides of the US-Mexico border, a region that continues to experience rapid population growth. The university's enrollment has grown along with the region's population, exceeding 20,000 students for the first time in academic year 2011-2012.

The Department of Computer Science at UTPA offers degree programs at the Bachelor and Master's levels. At the undergraduate level, the department offers two accredited degrees, the CAC/ABET accredited Bachelor of Science in Computer Science (BSCS) and the Bachelor of Science in Computer Engineering (BSCE), a joint EAC/ABET accredited program with the Department of Electrical Engineering. Computer engineering majors will select a hardware or software track, with some differing requirements for the two tracks. Graduates of the undergraduate degree programs usually achieve good entry-level industry positions with companies such as IBM, Intel, and American Airlines, or continue to graduate study. The department currently has between 500 and 600 undergraduate majors, split relatively evenly between computer science and computer engineering.

At the graduate level, the department offers programs for a Master of Science in Computer Science (MSCS) and a Master of Science in Information Technology. (MSIT). The MSCS program is intended primarily for students with an undergraduate degree in computer science or a related discipline, while the MSIT program targets students who are making career changes from other fields. MSCS students choose to complete either a thesis or a Master's project plus Master's final exam. The MSIT degree does not have a thesis option; all MSIT students must complete a project and pass the final exam. The final exam covers the required core courses of the respective degree program, and is offered once each semester. Many of the graduate students, and in general most of the MSIT students, have full-time employment in addition to being a student.

III. Project Framework

A. Participating Courses

The software project involved students in six different courses, some of which are cross-listed concurrently taught classes. The courses included both undergraduate and graduate students. Below, we describe the courses, their place in the computer science and engineering curriculum, and the general student demographics for each course.

- **CMPE 3340** and **CSCI 3340 Software Engineering.** The software engineering course is required for both computer science majors and computer engineering majors. Prerequisites for the course are CS II (the second programming course), and one upper-level computer science course. Students typically take the software engineering course during their junior year, since it is the prerequisite for the senior project course (see below). The CMPE and CSCI courses are cross-listed and taught concurrently. Enrollment has been climbing, nearing 40 students total (combined CMPE and CSCI) in recent semesters. The enrollment is split roughly evenly between computer science and computer engineering majors.
- **CSCI 4390 Senior Project.** The one-semester project course is required for all computer science majors, and is typically completed in one of the last 2 semesters before anticipated graduation. The student designs and executes the project with the supervision of a faculty advisor. There is a large degree of variation in the types of projects completed by students. Some projects are research-oriented, while others are application-centered. Department faculty members evaluate student projects, with the most weight given to the evaluation of the student's faculty advisor and the project course instructor.
- **CSCI 6315 Applied Database Systems** and **CSCI 6333 Advanced Database Design and Implementation.** The two database courses are typically taught concurrently. MSIT students take this course as part of their core requirements and the database course content is included on the MSIT final exam. MSCS students may take this course as an elective. The course covers a wide range of topics including entity-relationship modeling, the relational model, structured query language, query optimization, and recent developments in database technology. The course typically has an enrollment of 15 to 25 students. Students complete individual assignments as well as a group project. The prerequisite for CSCI 6315 is a leveling course in programming and software systems, and the prerequisite for CSCI 6333 is a course in algorithms and programming languages.
- **CSCI 6340 Advanced Software Engineering.** The graduate level software engineering is an elective course for MSCS and MSIT students. The course is required for the graduate students in the Engineering Management program of the Department of Manufacturing Engineering. On occasion, students from other programs (*e.g.*, electrical engineering) enroll in the class as an out-of-department elective. The course typically has enrollments of 15-20 students, and is a project-based course. Because the course must accommodate a wide range of students, the course prerequisite, a programming and software systems leveling course, may be waived in favor of instructor permission.

B. The Project

The project was to create requirements specification, analysis, system design, and object design for a museum tour reservation software system for a local historical museum. The education specialist for the museum contacted the computer science faculty for help with issues in the current software solution that emerged when the museum migrated to Windows 7. The old software is based on FileMaker Pro and is Windows 98 compatible, and withstood the migration to the new operating system. However, there were issues with loss of data, and mounting concerns about continuing to use a product that is far past end-of-life. The existing software solution also lacked several important features that would greatly improve workflow for the client, such as the ability to share and export data with the accounting officer and with museum executives, and automatic generation and emailing of confirmation letters to groups booking tours. Project teams were tasked with recreating the necessary functions from the old software and expanding the functionality to include desired features as prioritized by the client.

The client was extremely enthusiastic about the project, and was responsive to all questions. She met with the project faculty on several occasions and visited both of the software engineering classes during requirements elicitation. At the end of the semester, she attended the software demos presented by the graduate software engineering teams and helped to evaluate and provide feedback on the teams' solutions.

C. Project Design

When planning the project before the beginning of the semester, we organized the project according to our conceptual corporate structure. The two software engineering divisions (*i.e.,* the software engineering classes) were to be responsible for the design and development of the software solution, while the database division (students from the two graduate database classes) was responsible for the database schema and the API for the database management system. The part of project owner was to be taken by the senior project student, who would be the primary point of contact and communication with the client and with the division managers (the course faculty).

The plan for the two software engineering divisions was to have all the teams work on requirements elicitation and analysis in parallel. Upon completion of requirements, the project owner and software division managers were to create a unified requirements document by integrating the work of the two software engineering divisions. From that point onward, the design and implementation of the system could be modularized, with each team given responsibilities for a particular vertical slice of the system.

The database division would consist of one of the project groups from the database course. Their task would be to design a relational database and provide an API to the software divisions. At the beginning of the semester, the database division would work on the basic database design and design drafts of the API, while the software engineering divisions worked through requirements. By the time the software divisions had a complete picture of the application domain data entities to give to the database division, the database team could then proceed to fleshing out these preliminary designs with the details from the software divisions. The software divisions would, in turn, have the fully specified database schema and database management system API in time for system design and object design.

In the two software engineering divisions, teams were organized with team managers and assistant managers. In addition to coordinating activities for the teams, the team managers would serve as the point of communication with division managers and with the other divisions. Division managers would provide contact information so that teams could communicate with their counterparts in other divisions. The project owner was to have to role of primary point of contact with the client, ensuring that all divisions had the same information.

D. General Considerations

In constructing our project design, we were mindful that we had to address a disparate skills base among the students in the participating courses. The differences in student skills fall into two main categories, the differing needs of undergraduate and graduate students, and discipline-related skills differences among the graduate students (MSCS students, MSIT students, students from areas outside computer science).

Undergraduate students generally need more help from the professor than graduate students in areas such as time management and organization. They also need guidance in shifting their perspective from completing discrete individual assignments to working in a team requiring sustained performance over an extended period of time. This transition has a strong component of shifting motivation from external forces to internal ones. Graduate students typically have more project and teamwork experience than undergraduate students, but need support to take the next step to become independent thinkers and problem solvers. That step involves the notion that the professor does not have all the answers, or that there may not be a right answer at all.

We anticipated that the graduate students involved in the project would have widely varying skills and backgrounds, a reflection of the fact that the students come from several different degree programs and departments. Among the computer science graduate students, the MSCS students generally have stronger foundations in computing skills (programming, theory, mathematics) than the MSIT students, who are often pursuing the degree program as part of a career change from a different discipline or line of work. Many of the MSIT students have had only the one-semester programming and software systems leveling course before enrolling in the software engineering or database courses. The engineering management students may have no programming background at all, although they bring training and experience in engineering process that is far more extensive than engineering training among the computer science students. In general, the different backgrounds are a plus for the software engineering course, not a minus, since software engineering is an inherently multidisciplinary activity. The course faculty ensured that each project team contained a mix of students with different backgrounds and skill sets. More heterogeneous skills on project teams provide a more realistic simulation of project teams in the business world.

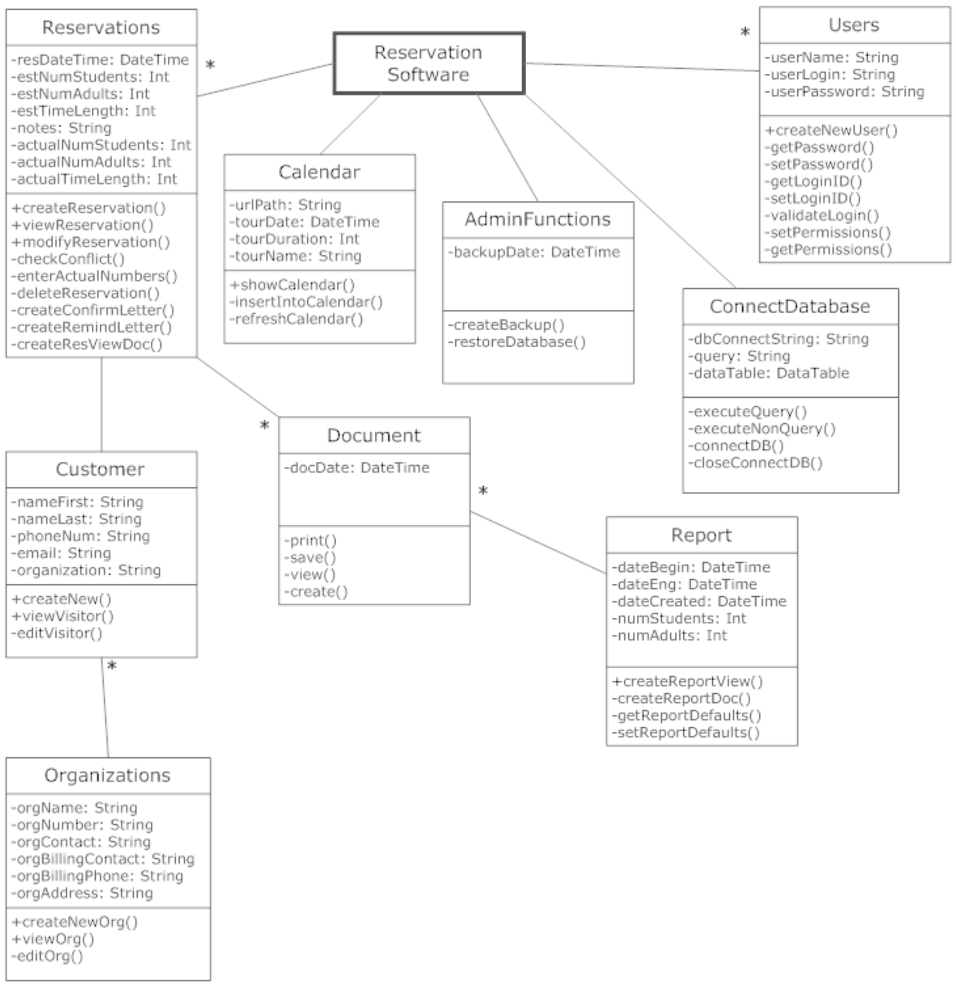

Fig. 1. Sample student work, requirements analysis. The software engineering teams produced formal documents, including requirements analysis, system design, and object design documents. The object model shown in the figure was created by one of the graduate student teams.

Because of the inherent complexity in the structure of the project organization, we attempted to ensure that the students would have a firm basis from which to begin the project. For this foundation, we provided the students with extensive documentation for the existing software that the client uses and installation files for that software product. We believed that these materials would give the project teams a big advantage in requirements discovery and analysis, since the materials provided a comprehensive view of the existing solution. Our thinking was that many of the essential requirements for the new software would be based on those of the old software. Our primary task as designers and developers would be to add the functions that were absent in the old software, while updating the system to a modern software design.

The software engineering classes typically do not end the semester with a fully realized software system. We did not anticipate that this semester would be an exception to that typical outcome. Our goal was to have a fully analyzed and designed system at the end of the semester, which could then be implemented as a senior or Master's project in the summer or fall. We considered this to be an acceptable goal, particularly when viewed in the context of the risk level of this project, which we evaluated as very low. The client has a working software solution, albeit an outdated one that lacks important features. The client was enthusiastic and flexible, and eager to work with the students. The client understood that we may not be able to deliver working software within one semester, and was willing to accept a longer timeline to delivery. Having such low project risk allowed us flexibility to focus on student learning instead of delivering software on deadline.

IV. Project Implementation

The database course project is divided into five development phases: design overview, creation of an entity-relationship (E-R) diagram, translation of the E-R diagram into a relational database schema, creating the database and loading it with data, and using a high-level programming language to create a simple application that demonstrates the capabilities of the database. Following these design phases, the students give an oral presentation and submit a written final report. As students complete each phase, they receive feedback from the course faculty and are expected to incorporate this feedback into the next phase of their project. One out of the eight groups in the database course worked on the project described in this paper.

From the beginning of the semester, the database division was faced with the challenge of synchronizing their development phases with those of the software engineering courses. Because the deliverables for the database design phases were due prior to the time when the software engineering divisions completed the requirements, the database division was directed to base their design on the software that is currently used by the client. They would then refine their design based on the requirements provided by the software engineering divisions. However, due to communication challenges and time constraints, these refinements were not made to the database design.

The software engineering courses at both undergraduate and graduate levels cover similar material, with differences in emphasis, degree of student autonomy, and expectations of product sophistication. The courses are primarily project-driven, with the software project encompassing most of the work the students do in the courses. The specific software engineering methodology for the project differs from semester to semester, since the emphasis is on process. Because of the size of the organization for the current project, the software engineering faculty decided to use a traditional approach, allowing for some iteration on previous phases as the project progressed. The courses placed a strong emphasis on requirements elicitation and analysis. The undergraduate students typically spend more time on requirements than the graduate students. This scheduling concession is made in large part because the undergraduate students are generally less experienced and skilled in interpersonal communication than the graduate students, and need more time to establish communication within the team and with the client. Teams produced a detailed requirements analysis, system design, and object design (Figures 1 and 2). Teams also worked on aspects of implementing a prototype of the system. The graduate students focused on designing and implementing prototypes for the user interface, while the undergraduate students focused on vertical slices of system functionality. We were unable to carry out the original plan for a unified requirements document, discussed in section III-C above, due to schedule misalignment and other time concerns. Each project team carried their own requirements analysis through the other phases of development. The graduate course culminated with a demo of the interface designs for the client (Figure 3).

The senior project course is highly individualized, with the plan for the project a collaborative effort between the student and the supervising faculty member. The project owner is a senior majoring in computer science. She has training in software engineering, has worked on several team projects and has completed an internship with the software development group in an international corporation.

V. Project Challenges

Even before the semester began, it was evident that the project would present many more challenges than anticipated. The challenges fall into three broad categories: class size, leadership, and communication.

When the three faculty involved in teaching the participating courses decided to pursue this ambitious cross-course project, we predicated the decision on the assumption that

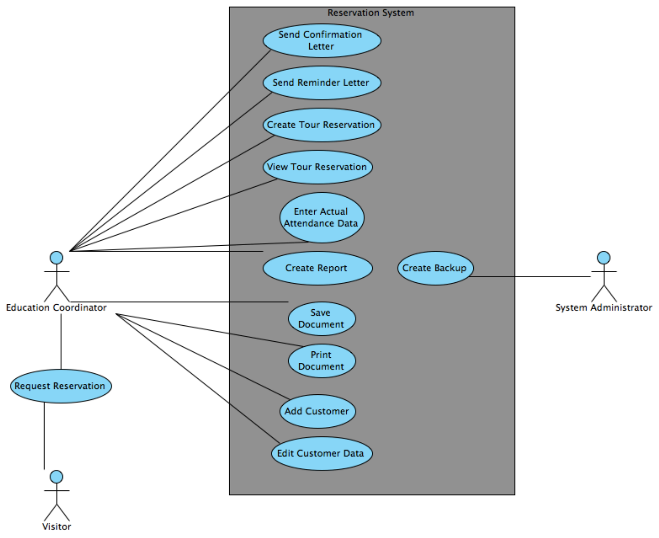

Fig. 2. Sample student work, requirements analysis. The software engineering teams spent the first part of the semester on requirements elicitation and analysis.

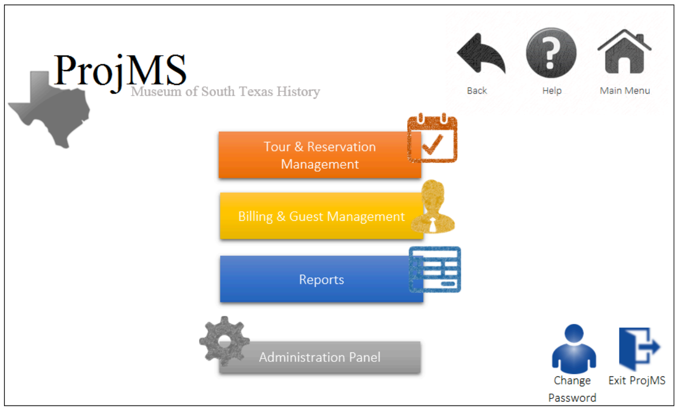
ProjMS
Museum of South Texas History
Back
Help
Main Menu
Tour & Reservation Management
Billing & Guest Management
Reports
Administration Panel
Change Password
Exit ProjMS

Fig. 3. Sample student work, interface prototype. The software engineering teams completed system design based on their team's requirements analysis.

the classes would be typical, in terms of student backgrounds and class sizes. However, unexpected events resulted in much larger than typical class sizes for the software engineering and database courses. The undergraduate software engineering course began the semester with an enrollment near 40, while the graduate software engineering class numbered 48 students at the start of the term The graduate database courses had a combined enrollment of 41. The number of students impacted the number and size of teams in each class. Instead of 3-5 project teams with 4-5 students each, classes now had 7-8 teams of 5-7 students each.

The number of students in the classes also necessitated some last-minute changes in process. Processes and methods that had worked well in the past for classes of 15 to 25 students simply would not scale to classes of nearly 50 students. For example, the graduate software engineering course usually relies heavily on in-depth interaction between the professor and each software project team, both through in-class conversation and ungraded previews of written documents. When the class suddenly had 8 teams instead of the usual 3-4 teams, that level of interaction became impossible. To compound the difficulties, the ballooning enrollments in the courses occurred less than 2 weeks before the semester began, making re-planning prohibitively difficult.

Leadership posed a significant challenge on two levels. Leadership from the student team managers was variable, ranging from strong to practically nonexistent. Broadly speaking, the team managers in the graduate software engineering class were more effective than their undergraduate counterparts. This is not an unusual situation, since graduate students, in general, have more experience and leadership skills to bring to the table than undergraduate students. Student team managers are always in a difficult position, since they are peer leaders, and have no true authority to hold their teams accountable for deadlines or quality of work. There was also a lack of leadership structure among the faculty division managers. We did not appoint one faculty member to take the role of project manager. The lack of a single project manager left a leadership gap that resulted in overall lack of coordination between the classes that became more pronounced as the semester progressed. Our implicit idea was to have the project owner take the role of project manager. However, even before the project began it became apparent that the student had neither the skills base nor the interest to take on a project management role.

The most profound challenge to the project was communication. There were communication issues at every level of the project organization. With no project manager and no formal meeting or communication plan, communication among the faculty division managers was sporadic. This poor communication exacerbated the scheduling coordination issues, since the individual faculty members had little information about the project schedule and progress in the other courses. The project owner (Senior Project student) communicated to some degree with the undergraduate software engineering class, but never communicated with the graduate students. This situation resulted in the graduate students having less information from the client than the undergraduates. Communication between divisions at the team level was almost entirely absent. The most serious gap in communication was with the database division. In the two software divisions, the large number of teams hampered communication between the team managers and their division managers. Division managers made efforts to speak with teams and answer questions during in-class team work sessions, and informally monitored division of labor and team process though weekly reports. While some team managers were comfortable reporting team issues to the division managers, others were reluctant to report problems until the issues escalated in severity. Teams had varying levels of internal communication as well. All teams had online communication tools that were provided by the university's online learning platform (Blackboard), and some teams chose to use other tools in addition to or in place of the Blackboard tools. Many teams used virtual meeting tools such as Google+ Hangouts for meetings outside of class time. Regardless of what tools were used, the underlying assumption is that all team members will use the tools, attend the virtual meetings, and complete assigned work. Unfortunately, that scenario appeared to be the exception rather than the rule among many project teams. Most teams experienced serious communication gaps, with team members missing meetings and deadlines, and failing to respond to messages sent by other team members or to complete tasks. On the whole, the undergraduate teams seemed more effective than the graduate students in establishing and following through on good communication between team members.

VI. Lessons Learned

We learned some valuable lessons in our first attempt at a project of this complexity. Two of these lessons are key: 1) the impact of planning and management roles played by the participating faculty members, and 2) the importance of a centralized communication structure.

The participating faculty need to be more proactive in planning and managing the project. Project planning must be more concrete. All aspects of the project suffered from a lack of schedule alignment between the project classes. Such scheduling issues can be resolved to a large degree with sufficient planning in advance. Good scheduling can still retain some flexibility in the timeline to accommodate differing levels of complexity within the project process. Some form of formal project plan would be helpful in this regard. Such a plan cannot be part of student learning in this context. Instead, it must be shepherded by the project faculty. To address the project management issues, one of the project faculty needs to take on the project manager role, in order to ensure that the project remains on track and on time. Our initial idea of having an undergraduate student assume the project manager role was naïve or overly optimistic. A strong graduate student could function as project manager, but only if the student is not involved in the project in another capacity, *i.e.*, if the student is not enrolled in one of the project courses.

Without question, communication posed the most serious challenges for the project. A strong, stable, centralized communication hub for the entire project is crucial. Although we established a good communication environment for the individual teams to use, there was no centralized communication for the entire project. A centralized communication structure would include communication at all levels of the project, as well as centralized access and distribution of project materials. The use of version control from the start of the project would be one aspect of the distribution system. The use of versioning at multiple levels (individual teams, project classes, and the entire project) would give transparency to the process as a whole, and enhance accountability for individual participants. The graduate software engineering class used version control for part of the project, but it was introduced too late, and students were not adequately trained in its use.

VII. Conclusion

Even with all the challenges, we accomplished our goals for this project, and learned a number of important lessons in the process. The students experienced the benefits and challenges of working as part of a large enterprise that is distributed through time and space. They worked with teammates who have a wide range of backgrounds and expertise. This experience provided students with the opportunity to gain in-depth domain knowledge in the area of software engineering or database design and development. Additionally, the students made good progress toward developing a product for our client.

We plan to continue working on the project for this particular client through individual student projects and additional course projects. A number of the students participating in the project this semester have expressed interest in continuing to work on the project for their senior capstone project or Master's project. The faculty member who lead the database portion of this project is considering using this project in the undergraduate database course. Her plan is to start with the products created by the software engineering courses in this phase of the project and use those products to guide the database design.

In a future semester we will consider the possibility of pursuing another project of this scale and will apply the lessons learned this semester to provide an improved experience for the students. We will designate one faculty member as the project manager and hold regular meetings between the division leaders in order to improve communication between the divisions. We will use a centralized communication platform to enhance the communication between project participants at all levels. Additionally, we will engage in additional planning prior to the start of the semester in order to ensure that the project schedules are better synchronized between the courses.

By combining the efforts of student projects in six courses to work on a project for a real–world client, we provided the students with a unique opportunity to experience the working environment of a complex organization. The lessons we learned regarding the importance of communication and coordination between the faculty and student participants can be used by other engineering faculty who are seeking to provide their students with a valuable learning experience through a large–scale group project.

Acknowledgment

The authors would like to thank our client, Mrs. Judy McClelland, Education Specialist of the Museum of South Texas History in Edinburg, Texas. Her passion for education made the project more interesting, and her boundless enthusiasm engaged and energized the entire project team.

References

[1] B. Brugge and M. Gluchow, "Towards production ready software in project courses with real clients," in *First International Conference on Software Engineering Education based on Real-World Experiences (EduRex)*, 2012, pp. 5–9.

[2] W. Grega, A. Kornecki, M. Sveda, and J. M. Thiriet, "Developing interdisciplinary and multinational software engineering curriculum," in *Proceedings of the ICEE07*, 2007, pp. 3–7.

[3] M. J. Sebern, "The software development laboratory: Incorporating industrial practice in an academic environment," in *Proceedings of the 15th Conference on Software Engineering and Training*, 2002, pp. 118–127.

[4] D. Suri and E. Durant, "Teaching requirements through interdisciplinary projects," in *Proceedings of the 2004 ASEE North Midwest Regional Conference*, 2004.

[5] M. Jazayeri, "The education of a software engineer," in *Proceedings of the 19th IEEE International Conference on Automated Software Engineering*, 2004, pp. 18–27.

[6] M. Herold, A. Ganci, B. Ribeiro, R. Ramnath, and R. B. Stone, "Work in progress–computer science perspectives on integration with human-centered design," in *ASEE/IEEE Frontiers in Education Conference (FIE)*, 2011, p. F4F.

[7] S. Dekhane and M. Y. Tsoi, "Work in progress–inter-disciplinary collaboration for a meaningful experience in a software development course," in *ASEE/IEEE Frontiers in Education Conference (FIE)*, 2010, pp. S1D–2.

[8] L. Jaccheri and G. Sindre, "Software engineering students meet interdisciplinary project work and art," in *Proceedings of 11th International Conference on Information Visualization*, 2007, pp. 925–914.

[9] A. B. Albu, K. Malakuti, H. Tuokko, W. Lindstrom-Forneri, and K. Kowalski, "Interdisciplinary project-based learning in ergonomics for software engineers: A case study," in *The Third International Conference on Software Engineering Advances*, 2008, pp. 295–300.

[10] A. Schaetter, H.-G. Koeglmayr, K. Blankenbach, and M. Nippa, "Interdisciplinary approach to software engineering education," *The Journal of Systematics, Cybernetics and Informatics*, vol. 7, no. 5, pp. 29–36, 2008.

[11] I. Richardson, L. Reid, S. B. Seidman, B. Pattinson, and Y. Delaney, "Educating software engineers of the future: Software quality research through problem-based learning," in *IEEE-CS Conference on Software Engineering Education and Training (CSEE & T)*, 2011, pp. 91–100.

[12] L. Johns-Boast and S. Flint, "Simulating industry: An innovative software engineering capstone design course," in *IEEE-Frontiers in Education Conference*, 2013, pp. 1782–1788.

[13] R. Bareiss and E. Katz, "An exploration of knowledge and skills transfer from a formal software engineering curriculum to a capstone practicum project," in *Proceeding of Conference on Software Engineering Education and Training (CSEE & T)*, 2011, pp. 71–80.

[14] D. J. Bagert, R. Dupuis, P. A. Freeman, S. Saiedia, Hossein, M. M., and J. B. Thompson, "Panel: Software engineering body of knowledge (swebok)," in *Proceedings of the 23rd International Conference on Software Engineering (ICSE)*, 2001.

[15] (2004. Accessed 12 March, 2014) Software Engineering 2004 Curriculum Guidelines for Undergraduate Programs in Software Engineering: A Volume of the Computing Curricula Series. [Online]. Available: http://sites.computer.org/ccse/

[16] J. Vallino, "What should students learn in their first (and often only) software engineering course?" in *Proceeding of Conference on Software Engineering Education and Training (CSEE & T)*, 2013, pp. 335–337.

[17] W. A. Lawrence-Fowler, L. M. Grabowski, R. H. Fowler, and G. Yedid, "Convergence of evolutionary biology and software engineering: Putting practice in action," in *Proceedings of 2013 Frontiers in Education Conference*. IEEE, 2013, pp. 356–361.