A Graduate Level Case Study Using a Real World Project: What Students Say They Learned

Chlotia P. Garrison*

Thurmond Building, Winthrop University, Rock Hill, SC 29733

Abstract: Allowing graduate students to develop a real world project for actual customers provides an opportunity for students to experience the benefits of following basic software engineering principles. Most universities now offer a course in software engineering and many information technology students must complete a class project in conjunction with their degree program. One reason businesses often struggle to implement a software process improvement program is that many practicing professionals have never experienced the complete software lifecycle and discovered how software engineering principles aid a real project. This paper presents the lessons learned by students in a capstone graduate software development course when required to develop an intensive real world project for real customers. Students had to solve the same type of problems encountered by practicing professionals, developed an appreciation for following software engineering principles, discovered the importance of good team communication and appreciated developing a product for a real customer.

INTRODUCTION

In addition to teaching theory, a goal of the education process should be to develop students that can succeed in industry. Students need to learn not just the theory but how to properly apply software engineering tools, techniques, and principles. Hoxmeier and Lenk [1] identify three factors that contribute to Information Systems (IS) graduates not being able to properly apply theory in practice: lack of public exposure to the final product, lack of complexity in class projects, and lack of inter-disciplinary project teams.

Information Technology (IT) and other disciplines often use class projects to help improve student learning. This article presents the results of a three-year period of having graduate students develop a complete project for real customers. The projects all produced products for real customers which provided public exposure, were more complex than the typical undergraduate class project, and often included students from multiple disciplines. Thus a class project developed for real customers can remove the obstacles to applying theory in practice identified by Hoxmeier and Lenk [1]. Students answered questions at the end of the semester about the project. Student feedback is that both students who have not yet worked in the software field and practicing professionals both appreciate the real world project and rigor of the course. This article presents an analysis of student answers that reveal some common lessons learned and areas of heightened awareness; the answers indicate these students will be better able to properly apply theory in practice.

*Address correspondence to this author at the Winthrop University 316 Thurmond Building, Rock Hill, SC 29733; USA; Tel: 803-323-2470; Fax: 803-323-4600; E-mail: garrisonc@winthrop.edu

INDUSTRY OBSERVATIONS

In October 2003, the author's former employer became only the second government organization to achieve a Capability Maturity Model®) Level 5 rating, the highest level of software maturity as defined by the Software Engineering Institute. The author was present at the inception of the organization's journey and experienced many of the challenges of instituting a software engineering process. One critical challenge to implementing software engineering principles is that experienced software developers who have been operating in an environment where there are few formal processes often strongly resist change to a more structured environment. This type of resistance, however, was also present among more recent college graduates who had studied the principles of software engineering but had never experienced the benefits. Both groups saw the change to a structured and disciplined approach as a loss of independence.

Discussions with graduate students working in the software industry indicate that many of their employers are not practicing the software engineering discipline. Some organizations ignore company standards, others have standards that are ineffective, and some have few or no defined processes. When professionals themselves recognize the benefit of software engineering principles, then it is much easier to successfully institute a process improvement program and, if necessary, to convince management of the need for such a program.

CLASS PROJECT AND ACTIVE LEARNING

One method to help students learn to properly apply and recognize the benefits of software engineering tools, techniques, and principles is the active learning available through class projects. Multiple researchers have studied and defined

1874-107X/08 2008 Bentham Open

active learning. Norman and Spohrer [2] state that incentive to seek new knowledge to solve a problem increases learning. A review of the research by Prince [3] defines three types of active learning: collaborative, cooperative, and problem-based. In collaborative learning students work together; in cooperative learning students cooperate but with individual assessment. Problem-based learning, which is often collaborative and cooperative, presents students with a problem and learning is frequently self-directed [4], [3]. Laware and Walters [5] distinguish between problem-based learning and problem-centered learning, asserting that problem-centered learning is more structured. A class project that requires students to develop a software product can include all these elements making possible the combined benefits of the various types of active learning.

The many benefits of active learning have been identified by researchers in many disciplines and include developing a more positive attitude in students, fostering a deeper approach to learning, helping students retain knowledge longer, increasing students' ability to organize, plan and execute, enhancing problem solving skills, enhancing students' academic achievement, interpersonal skills, and retention [3], [4], [5]. Edelson *et al.* [6] developed a software tool to aid students in doing collaborative, open ended activities. They found that contrary to what they expected, students generally did not recognize the benefits the software would provide before they used it. Students only used the software when there was a clear advantage to them, such as being required to perform an activity using the software as a part of their grade, or allowing them to contact students with different schedules or in different locations. Lave [7] views learning as social practice and questions the traditional understanding that separates formal and informal education. His research into apprenticeships revealed that traditional teaching methods are not essential and also insufficient to produce learning. Holt *et al.* [8] advocate a curriculum that brings together theory and practice through a cooperative program that places the student in a real business environment. The results of their research show that the value of the program to the students varied widely based on the specific placement of the student. Holt *et al.* [8] further identify quality as the interactions between the student, faculty, and the profession. Students involved in research on pair-programming by Williams and Kessler [9] noted the following collaborative benefits: increased quality as a result of continuous review by their partner, the added assistance of the partner to help solve problems, and the self imposed "pair-pressure" of not wanting to disappoint their partner. Tanniru and Agarwal [10] found that students in a 4-semester program that uses projects to link theory and practice had greater success on more complex problems because they liked being able to meet the challenge. Schuldt [11] in contrasting the benefits of simulated and real-world projects in IS education state that both types of projects help sharpen the students' communication skills.

A class project that produces a product for an actual customer is also a service-learning opportunity, which offers additional benefits. The Corporation for National and Community Service [12] describes service-learning as an opportunity for students kindergarten through university to become involved with their communities by combining service projects with classroom learning. They assert that service-learning programs can improve grades, increase attendance in school, develop students' personal and social responsibilities and promote a sense of caring for others. Bringle and Hatcher [13] add that service-learning increases understanding of the course content and increases appreciation for the discipline. Fox [14] states that a real-world project increases the new graduates' confidence that they can succeed in the business world, and employers feel these students are better prepared than students who have not had the experience of developing a project from inception through fielding. Magboo and Magboo [15] use the assignment of a real-world prototype project to benefit both the university and the students. The university receives a working prototype that supplements a constrained budget and the students gain a sense of achievement and an appreciation for various software engineering principles. Harris [16] states that a systems project for a real customer gives the students opportunities to improve their written and oral communication skills, areas that the IT community consider critical for success.

While the information on real world projects is increasing, additional studies add to the empirical research. Most of the research and case studies address undergraduate students. The increased challenge possible in a graduate service-learning software project should mean greater success and an increased appreciation for the discipline. This article adds to the body of material available in general, adds the component of being for graduate students, and includes what the students (including practicing professionals) learned about software engineering through developing a real project.

PROJECT DESCRIPTION

The Software Development course in the Master of Science in Software Development program at the author's institution calls for "an extensive and intensive project involving all aspects of a software development project including teamwork, requirements specification, design, configuration, coding, testing, quality control, and evaluation" [17]. The university offers the software development course annually. The presented results cover three offerings of the course. Each class required students to develop a real product for use by real customers. The instructor informed the customers that because the software was being developed as a class project the project had to be completed by the end of the semester, success was not guaranteed, and no maintenance was available once the semester ended. The completed software is loaded in the customers' environment and they have free access to modify the software once the project ends.

Project 1 developed a course scheduling system for the College of Business Administration at the author's institution. Prior to the project, department chairs and the associate dean used a time consuming manual process to schedule courses using a series of meetings and e-mails to resolve conflicts over time slots, sequencing and available rooms. The resulting system allows department chairs to enter course scheduling information and to immediately see information already entered by others. The system generates warning flags for conflicting entries and allows authorized users to perform maintenance functions and view and print a variety of reports. The system has now been in use for 4

years. The customers were extremely happy with the finished project. One of the customers was a computer science professor and stated that he had not expected the many capabilities of the resulting system.

Project 2 created a resource directory tool for a leading non-profit social services organization. The interactive database provides the agency the ability to have a complete, up-to-date and searchable directory of area agencies. The directory provides an efficient way for one agency to refer clients to other agencies for additional services. The directory also provides the capability to match volunteers with service agencies using interests, skills, and expertise. Prior to this project, a directory of agencies that provide social services in the county did not exist. The customers were extremely pleased with the product. The project has been in use for over 3 years and the customers have made it accessible through the Internet. At the time the project was developed, the agency did not have a website.

Project 3 was developed for a local church. The goal of the project was to provide a functional website for use by the general public and the members of the church. In addition, the website provided a secure area for members. In the secure area, members can view and maintain personal and membership information. Members are able to view information about other members and generate reports of selectable information. This project was not a success because of a problem in updating the site. Though the members-only area was updatable using the developed database, the customer could not easily update the publicly available pages. The public pages were created using the wizard feature of Microsoft FrontPage and would revert to the previous page when they were updated. A subsequent class redid the project and it has been in use for over a year.

Because of the small class size, the entire class acted as the team for each project. The team size was 4 students for projects 1 and 2 and 3 students for project 3. The instructor acted as project manager and each team selected a team leader. Every student was individually responsible for writing one formal document with team input: Software Requirements Specification, Project Plan, Software Development Plan, Test Plan and Summary, or User's Guide. Each project was a major endeavor that required most of the semester to complete, required significant effort on the part of the students, and required students to learn some technology or capability previously unknown to them. The instructor introduced the projects at the beginning of the semester with the deadline coinciding with the last or next to last week of class. Lectures and class discussions focused on techniques for developing solid software.

The goals of the class project are to provide the students with experience in the complete software development lifecycle, help them gain an appreciation for following software engineering principles, give students experience in working with a team, communicating with a customer and preparing formal documentation, and help prepare students to succeed in industry. The class projects are also an opportunity to provide a service to the community. Specific software engineering principles emphasized by the project and the class discussions included requirements elicitation, the importance of documentation (especially requirements, design, and testing), configuration management, peer reviews, customer involvement throughout the lifecycle, risk management, and project planning.

Students had to manage the projects to be completed by the end of the semester. Students review requirements to determine their internal project schedule. The students inform the customer at the beginning of the project and throughout the lifecycle of any requirements they will not be able to complete. Some requirements can not be completed on time because the customers fail to provide information in a timely manner. Some teams have had to refuse to add new requirements and have asked the customers to prioritize requirements because of the mandatory project end date. Because the customers are kept informed, have sometimes been slow to provide requested information, and are receiving a tool they did not have previously, customer feedback has been positive even if all initial requirements are not completed by the end of the semester. Customer feedback indicates they would recommend the student teams to other organizations.

Students performed the following activities in varying degrees: elicited and documented requirements, developed a project plan with schedule, designed the software, performed peer reviews, and communicated with the customers throughout the project. Students implemented the software, developed a test plan for use by the team and the customer, tested the software, fixed software problems, developed a user's guide, and installed the software in the user's environment. Students gave periodic status reports throughout the semester.

LESSONS LEARNED BASED ON STUDENT FEEDBACK

Students answered a series of questions at the end of the semester about the project. Students were asked to identify tools, techniques and principles they thought applicable to their project whether used or not and why. They also identified techniques and principles that were not applicable to their project and explained why they were not applicable. Students provided lessons learned for the class project and for their current or future career. An analysis of their free-form answers reveals some common themes.

Teamwork

The second lesson presented in the Software Development course is on successful teams. Students are asked to identify problems that can occur with software teams. Students readily identify lack of technical ability, personality conflicts, communication challenges, and failure to execute. All these are potential real issues that a team may encounter. The class then focuses on team formation, methods of conflict resolution and success characteristics. Ideally, team members are selected based on the knowledge, skills, and abilities needed to accomplish the project. However, the class roster determines the project team. The instructor assured the students that this is not atypical of industry. Depending on project priority, an organization may select team members from available employees instead of the best qualified for the project. The instructor informed students that there is rarely equal ability on a team. In order to have suc-

cessful teams, assign individual team members tasks based on ability.

Team members must research and bring in examples of team building exercises, and the class performs one of these exercises at the end of the class. Two exercises that the classes have conducted successfully follow:

1. The first requires a box containing items such as paperclips, sticky notes, sticks of gum and batteries. The instructor takes off the lid and all the members look inside the box for 60 seconds. After one minute, the lid is replaced and the members individually list the items they can recall. The members then compare results. It is interesting that some members see some items and some another. Alternatively, one member sees only one of an item while another sees more. The instructor then removes the lid and members see for themselves what they missed. The team saw and discussed the fact that the team is able to achieve better results than any one individual. Even though some remembered or saw more than others saw, no one remembered everything.
2. Another selected exercise emphasized the importance of communication. The instructor read the instructions while each member took a single piece of paper and closed their eyes. The students could not ask. The teacher instructed the members to repeatedly fold and tear the end off their paper. Once all instructions were complete, everyone opened their eyes and their sheets to reveal the results. No paper looked the same. When asked if they thought the results would have been different had they been allowed to ask questions one student stated that he didn't have any questions but he may have had feedback been given. This exercise not only demonstrated the value of two way communication but the fact that we can think we are doing everything just as expected and still be doing something different than desired.

In addition to the application, the team building exercise serves as an icebreaker for the team members and the instructor.

Comments reveal that students learned the value of a good team and the problems associated with a dysfunctional team. Students saw both the advantages of good communication and experienced the adversity of poor communication. Some teams had to overcome cultural and personality conflicts before eventually becoming a functional team.

Representative comments from students follow:

- *Teamwork was the biggest factor in the project because within the group there was a variety of domain knowledge. The team used many of the suggestions on building a successful team that we discussed in class, such as planning, clearly defining roles, clear communication and a reasonably balanced participation based on the knowledge of the team member.*
- *I've learned the value of solid, dedicated teamwork. I've been on unmotivated teams that did not know their roles. This team worked hard, worked very well together, and understood our individual roles, as well as our team objective.*
- *Most importantly, I learned that it is very important to respect your team members. (Even with different personalities and disagreements) it's still very important to respect others and their views.*
- *I learned that it is extremely important to have clear open communication when working on a project team. When there are several people working on the same product, it is essential that communication takes place whether it is to let the team know of our status, issues that you may have encountered, etc.*
- *I also realized that to get the full benefit of collaborative design, we really had to learn how to communicate effectively with each other. We definitely had some communication issues in our group… After a certain point, we were able to meet some where in the middle and communicate more effectively.*

Preliminary Research

Each project required students to use some tool or technique they were not familiar with or use it at a deeper level than previously used; an example of problem-based learning. Each project had a password feature that the students had to learn to implement in the application or the customer's environment. Project 1 team members had to install the product on the school's intranet and incorporate the schools authentication system. Project 2 team members had to investigate and learn to use the security features of Microsoft Access. Project 3 team members chose to use two software products without investigating their integrated security capabilities. When they tried to implement the security aspect of the database with the web application, they were unsuccessful. They had to switch tools near the end of the project. This issue led to the project being in an unacceptable state relative to updatability when provided to the customer. All team members commented on the need to do preliminary research and not assume a product's capability.

- *Find out the limitations of a certain software before we try to use it. I learned that a little research in the beginning can save you a lot of time in the end.*
- *This project taught me a valuable lesson in NOT assuming a software package will work according to its requirements. It also made me aware that we must fully understand all a project entails so that you can choose the resources accordingly.*
- *We should have done some preliminary research to find out the best tool to use from experts in the field.*

Prototyping

Prototypes are a valuable resource in software development. Prototypes can be useful in the requirements elicitation, design, implementation, and testing of a product. Each class discussed the benefits of prototypes as a tool for increasing the quality of software. Though the instructor encouraged the use of prototypes, the teams were not required to develop one. Two of the projects developed a prototype and realized the benefits. For example, the project 1 team developed a skeleton database and sample reports. The prototype helped to clarify and correct requirements, revealed

conflicting requirements and educated some users on the potential of the product. During the prototype demonstration a conflict in how the customers wanted the data filtered for a report surfaced; an option was included to allow users to filter the data multiple ways.

The project 2 team chose not to develop a prototype but later recognized how a prototype could have enhanced the project. Students learned that a prototype can help in eliciting requirements and facilitate user interface design. Students from all three projects commented on prototypes.

- *I do not think we would have done nearly as well gathering the requirements if not for the prototyping. I think that prototyping actually yielded better information than the interviews in some respects.*
- *I learned a very valuable lesson about prototyping. It is a very good way of receiving gradual sign off on a project. At the end, questions were raised about whether the customer would like certain design features. Had we used a prototype, we would have known the answer to those questions earlier in the project life cycle. While installing the product for user acceptance testing the customer immediately noted that they would like to see the phone number displayed. I would have known this prior to testing if our team had implemented prototyping and incrementally released the product.*
- *One thing that would have prevented us from finding certain faults in design so late in the process may have been to do a prototype for ourselves to determine how we wanted the system to navigate from one screen to the next.*
- *We developed a prototype so the customer could get a preliminary idea of what [the product] would look like.*

Peer Reviews

Effective peer reviews have great potential for improving the quality of software. Unfortunately, experience and research have shown that many reviews are ineffective. Studies by Porter & Siy [18] show that factors such as reviewers, authors, code units, team size and number of sessions can affect review effectiveness. In discussing the Team Software Process, Watts Humphrey [19] states that engineers stopped performing quality reviews without the constant oversight of the quality manager. Pflegger, Hatton, and Howell [20] discuss the pitfalls of the review process and how such factors as planning, individual preparation, reviewing speed, and psychological factors affect reviews.

The potential for significant resources being lost to ineffective reviews partly explains why some organizations do not practice them. An informal survey of students reveals that even many software professionals have never had their code reviewed. An early class is dedicated to defining the various types of peer reviews and presenting how effective reviews can be when done properly. In addition, the instructor presents the execution problems that are evident from industry experience. The instructor gives the students practical guidelines for the actual conduct of a review meeting, forms for recording findings, and the instructor acts as moderator for reviews conducted during class. Students are cautioned that proper preparation is required for the review to be successful. The most critical guideline for the review meeting is for reviewers to address the product and not the author: the *product* does not, instead of *you* did not. The moderator repeatedly reinforces this technique during the actual review because it takes effort to remember. The students and instructor review the Software Requirements Specification and the User's Guide during class. The instructor advises students to review other documents and the code outside of class. Evidence of these reviews was not required. Students realized that peer reviews were helpful in discovering and avoiding problems, saw where additional code reviews could have prevented some problems, and found that peer reviews were an opportunity to learn from the knowledge of others.

- *The peer reviews were quite helpful because it enabled those with more domain knowledge a chance to share it and for everyone to have input in the decision at hand. The reviews also took the 'egos' out of the process as we concentrated on the application and not on the author.*
- *If more communication had been in place, or more status updates, or even a code review, this mishap (search functionality implemented differently by different team members) could have been prevented.*
- *Having feedback on what you are working on is also important. If you can get the perspective of one or more people, it can benefit your work.*
- *While we occasionally do design reviews [at work], we typically skip code reviews. I would recommend the implementation of code reviews.*
- *While I think that the egos of the developers would initially make peer reviews painful, I see several aspects of our current application which would have benefited from an objective evaluation.*
- *Peer reviews were a tremendous help in driving out issues and problems during the project.*

Requirements

The instructor presented the proposed projects to the class with an overview of their purpose. After the initial customer meeting, the students interacted with the customers apart from the instructor to refine requirements for the system. The teams used interviews, prototypes, storyboarding, and use cases to elicit requirements. Students had to properly interpret requirements and cope with conflicting requirements. Because of time constraints, students had to limit the time for allowing new requirements and had to decide if they could meet all requirements. Often a trade off has to be made in business between providing everything a customer desires and the economical benefits. Students gained experience in overcoming the challenges of the requirements elicitation process, discovered the value of having documented requirements and the shortcomings of not following the requirements.

- *We benefited from a well-defined requirements document. We were able to nail down the requirements through good communication with the customer, and take advantage of prototyping.*

- *The requirements document was used the most. However, we still had people that made changes that were contrary to the requirements.*
- *It's so easy to jump into developing before the actual requirements are complete, but to prevent as much re-work as possible it may not be a bad idea to have a solid set of requirements before beginning the development phase.*

Design

In class we discuss the importance of design and elements of good design. Students perform design and implementation outside of class; however, we devote some in-class time to design considerations. A separate design document was not required; the project plan template included a section for presenting high-level design. Some teams used the Unified Modeling Language to help document their designs. A real project for real customers provided an excellent opportunity for students to experience the shortcomings of skipping or glossing over design as well as the rewards of a good design. Students learned that a good design creates a shared vision of the product which helps to prevent faults and also learned to appreciate the benefits of a modular design.

- *The majority of faults in the product are a result of poor work during the design phase. A clear vision for how the application would behave was not shared among team members, particularly with the search feature.*
- *I also learned that it is extremely important to have a good design in place before actually trying to develop any software. One problem we ran into was everyone was eager to start coding but we did not take the time to sit down and talk about, communicate, what the application should look like.*
- *I also learned why collaborative design is so important, though I fully realized this late in the process. There were faults that one person may have had on the module that they worked on that they didn't see as a fault until another team member performed unit testing on their module.*
- *Modularity – once the application was complete I was able to identify areas* of *modularity that had been successful. For example, the "Get Details" button executes a function to retrieve an individual agency. When the search function was re-written we had only to attach the "get details" function. This experience gave me a greater appreciation of the concept.*

Risk Management

The instructor presented a model for risk management and cautioned the students to both assess and control risks. The instructor presented categories of risks and provided a form for identifying and tracking risks. For example, students select a status for each risk: research, accept, watch, mitigate, or close. The students review the risks with the instructor during the periodic status reports. Students learned they must not only identify and track risks, but also develop viable mitigation plans.

- *The team performed risk assessment and successfully identified risks of events that actually occurred, namely the lack of knowledge of the development environment. A stronger mitigation plan could have lessened the impact.*
- *We used risk analysis to identify problems that arose during the implementation and design phase. Risks were identified but not mitigated very well.*
- *We also had to analyze risks of our predictions on the system as we developed a new system no one had seen before.*

Planning

We address the need for planning throughout the course. The review of the project plan template and regular status meetings emphasize planning. One of the items in the status report is *Goals for Next Review.* Teams have used Microsoft Project and a Word table to document the project schedule. The lessons on testing stress the need to plan testing well in advance of the testing phase. The instructor emphasized that the purpose of documentation goes beyond the document to planning a course of action. Students recognized that planning can make a task easier and help keep the project on schedule.

- *I have worked as a software developer for over 10 years and this project was the first time I was ever involved in the creation of an actual test plan. This is something I will recommend on future projects in my career.*
- *We tested our system over the weekend and were able to identify, and correct several faults by following a well-planned test plan.*
- *I learned that planning and organization is essential to completing a project on time. There were times that we got unorganized and it led to some lost time and inefficiency. While not a complete cure to help insure that a project will be completed on time, planning and organization will definitely make things easier.*
- *Our team not only benefited from giving dates to our stakeholders, but from setting well planned dates for ourselves.*

Configuration Management

Configuration management is presented as one of the elements needed to develop solid software. We discuss several aspects of configuration management including change management, identifying configuration items, status accounting, control boards, and tools. Because the students primarily used machines not on the campus network and did not install the software in the customers' environment until later in the project lifecycle no formal configuration management tools were available. The students had to develop a technique that would work for their team and their project. One team had a designated configuration manger that made all updates to the database on the website used for development. The configuration manager performed manual version control. Other team members did have access in case the configuration manager was unavailable. Another team used e-mail and version numbers to perform configuration management. Students acknowledged the value of configuration management to the project.

- *The technique that we used and I thought worked very well and is very important in developing solid software, is the use of the configuration management process. This process ensures that the project team members are not stepping on each other's toes and that the latest version of the software is what gets implemented. We used a 'check in' 'check out' method via an e-mail system and made sure the naming convention of the database included the date & version number.*
- *We did do configuration management, which was difficult given the nature of this project. Definitely a necessity or software could have been overlapped.*

Appreciation for the Software Engineering Process and Principles

The instructor provided templates to assist students in formally documenting the requirements, project plan, test plan/summary and user's guide. Students performed configuration management, peer reviews, risk management, and project tracking. The instructor provided standards and forms for risk management and peer reviews, and emphasized the importance of following a software development process. The instructor also presented such best practices as hazard analysis, error-handling design, and maintaining historical records. Student comments reveal that many gained an appreciation for the discipline of software engineering, which will make it easier for organizations to implement process improvement efforts. Both working software professionals and students who had never worked in the industry expressed new appreciation for following a disciplined process.

- *I have developed a new appreciation for a more formal and systematic approach to software development which has been missing throughout most of my career.*
- *I've learned that it's very important to follow the development life cycle as much as possible. It makes life a little easier when you work on a project in phases as opposed to just doing whatever is necessary to get the project completed.*
- *Overall I am happy with the experiences gained in the class and will continue to take tools and techniques into my workplace.*

Fault Prevention

The lessons in the software development course focus on developing solid software. There are lessons on hazard analysis, designing to prevent and tolerate faults, and using peer reviews to discover and avoid faults or defects. The formal peer reviews of the requirements document conducted by the team with the instructor helped to prevent many faults. The prototype of Project 1 highlighted a misunderstanding of requirements early in the life cycle, preventing the students from implementing the wrong functionality. In-class discussions of design and design alternatives increased the amount of error handling in the final products. Students also recognized that a good design, a prototype and solid requirements can help prevent faults.

- *I do believe that this particular issue could have been prevented if we had good design and prototyping in place. This would have allowed everyone on the project to have a visual idea of what this piece of functionality should look like on the screen and how it should behave.*
- *It's so easy to jump into developing before the actual requirements are complete, but to prevent as much re-work as possible it may not be a bad idea to have a solid set of requirements before beginning the developing phase.*
- *One thing that would have prevented us from finding certain faults in design so late in the process may have been to do a prototype for ourselves to determine how we wanted the system to navigate from one screen to the next.*

General Lessons

Students expressed the following general lessons learned: the importance of documenting code, the need to communicate, and the value of good estimates.

- *A weakness I have is not documenting what is going on and what I am doing. The project has shown that documentation is important because there were times where we were called on to debug each others code without a complete knowledge of what that code did. Documentation would have sped the process up tremendously. As the project progressed, I began to get in the habit of documenting what changes were made for future use. In the job I will be starting I know it is very important to document all the work that I do.*
- *One lesson that I learned and I will definitely apply to my current & future career is to speak up when you have an issue with your part of the project. No matter how big your ego, or how you think people perceive you, speak up because you never know who may be encountering the same issue or better yet, you don't want to delay the project because you did not open your mouth.*
- *One of the largest lessons is the importance of communicating and communicating well. I will be required to be able to use many methods of communications with fellow employees and the clients that I work with. Knowing how to handle a particular situation and communicate effectively is definitely something that I have seen and will always need to work on.*
- *Learning and having good estimates are important. I can see how important it is to learn from past experiences and judgments and to have that help you in the future.*
- *Although our time to complete this project was a little short, I think it's very important to follow the tasks & dates in the software project plan as much as possible.*

INSTRUCTOR LESSONS LEARNED AND RECOMMENDATIONS

- Make sure students give the customer specific timelines for providing answers or artifacts. Students often asked that customers provide artifacts or complete reviews ASAP. But customers, as with anyone else, tend not to act until there is a specific deadline. Providing the customer with specific dates also improves customer understanding if students can not implement some features because of time constraints.

- Identify potential projects prior to the start of the semester and assign the project within the first few class meetings. The upper level graduate classes at the author's university have recently transitioned to once a week classes. This makes it even more important to assign the project in the first or second meeting.
- Bring the customer and the students together very early in the semester. The instructor should attend this meeting to become more familiar with the system and help students elicit requirements.
- In a graduate class, some students may have extensive experiences and strengths that particularly benefit the project. The instructor must monitor the tasks performed by students to keep one student from carrying the project. Watch for unequally weighted tasks in the project plan. Remind students that learning and performing new tasks are a goal of the project.
- In addition to periodic status reports, have the students present the developing product to help the instructor assess the progress of the project.
- It is best to have students develop a prototype. A prototype helps the customers refine their requirements and can help to identify any technical challenges early in the project.
- The instructor, in addition to the customer, must perform acceptance testing before the project is complete. The customers were often so impressed with just having a product, they did minimal testing. Early instructor testing allows time for the students to address errors.
- The instructor should serve as an expert reviewer of the requirements document. Though the customer is asked to review the requirements document, their reading of the document is sometimes superficial.
- Encourage students to maintain ongoing communication with the customer. Students need to inform customers of the product's progress, significant problems being encountered, what they need to provide, and any features the students will not be able to implement.
- Identify multiple projects if possible and allow students to select the project they will develop.
- Make sure the customer is aware of the risk involved and that there will be no maintenance support for the product.
- If time permits, have students prepare a maintenance document in addition to the User's Guide. The maintenance document should provide subsequent developers knowledge about the project, features not implemented, and recommendations on how to fix any known defects or weaknesses.
- Have students track and provide, during status meetings, the number of hours they have worked on the project. This information is helpful for the instructor and the student. Students can use this information to improve future estimates. The instructor can use the hours to help gauge work distribution between team members, determine the magnitude of the project, and report service learning hours to the university.
- Conduct peer reviews. The instructor served as a reviewer and the moderator for the requirements and the User's Guide reviews. Most students, even working professionals, have not experienced a peer review of their work. Provide formal guidelines for the review and emphasize that students should address the work product and not the author.
- Have students rate each other and themselves. A technique similar to one the instructor learned at a conference has proven successful. Give the students an uneven number of points to divide among the team. The students must assign whole numbers to each member, including themselves. This forces the students to identify those that have contributed the most to the project. The distribution of points has been surprisingly consistent among team members, even those given the lower number of points. The instructor assigns a single grade to the project and uses the student assessments to assign a project grade to each student.
- Identifying potential projects can be difficult. If the university has a service learning center, make them aware of your course and the type of projects you are seeking. The Small Business Development Center and colleagues are also a source for projects. Make people aware of the opportunity for no-cost software development for non-profit organizations and others will often suggest projects to you.
- A real project by graduate students requires less technical problem solving on the part of the instructor. Graduate students are expected to learn concepts independently. Also, working professionals often bring related experience, code, or assistance from co-workers to the team.

CONCLUSION

Dromey [21] lists several factors needed to achieve software quality by preventing defects. These include proper use of effective process and product standards, formal inspections, identification and management of risks, prototyping, and good requirements elicitation. Using a real project provides students with the opportunity to practice each of these factors in a business environment. Requiring students to complete a real project with real customers also provides students with experience in skills sought by today's Information Technology Chief Information Officers.

Hoffman [22] identified troubleshooting, conflict resolution, interpersonal communication, project management, business skills and systems integration as perceived areas where today's college graduates fall short. In developing the class project, students practiced troubleshooting skills as they tested the product and determined not only the source but also the solution to defects. Interacting with real customers allowed the students the opportunity to experience and resolve the challenges of conflicting, changing, and unknown requirements. Students gained experience in interpersonal communication skills as they interacted with the customers and the other members of the team in unscripted and sometimes conflicting situations. They increased project management skills as they managed the project from initiation to fielding. Students in addition to the team leader ex-

pressed that they had learned management skills in developing the project. Students increased their systems integration skills as the team members integrated the components of individual members to form the system. The required documentation also provided students with practice in written communication.

The results experienced in requiring a real world project with real customers have been very positive. The students experienced challenges that would not have been as evident had the instructor been the only customer, and the project been artificial and of reduced scope. The students felt an obligation to provide a good product to the customer and worked extensively to overcome all challenges. The students enjoyed working on a large, purposeful project, achieved a sense of accomplishment, and gained a specific achievement for their resume.

The use of a real project not only benefited the students that have never worked in the software industry but also the working professionals who have never followed a formal process or formal standards or whose work has been segmented and they have never had the opportunity to experience the entire software development life cycle on a real project. Because these students now recognize the benefit of software engineering principles it will be much easier for organizations that employ them to successfully institute a software process improvement program. These students are also more likely to become advocates for initiating such programs. The challenges and successes experienced in a real project make these students eager to join a mature software organization and better equipped to apply software engineering principles in practice.

REFERENCES

[1] J. Hoxmeir, and M. Lenk, "Service-Learning in Information Systems Courses: Community Projects that Make a Difference", *J. Inf. Sys. Educ.*, vol. 14, pp. 91-100, Spring 2003.

[2] D. A. Norman, and J. C. Spohrer, "Learner-centered education", *Commun. ACM*, vol. 39, pp. 24-27, April 1996.

[3] M. Prince, "Does active learning work? A review of the research", *J. Eng. Educ.*, vol. 93, pp. 223-231, July 2004.

[4] "Active Learning: Reviewing the Research", *Teach. Prof.*, vol. 19, pp. 3, June/July 2005.

[5] G. Laware, and A. Walters, "Real World Problems Bringing Life to Course Content", in Proceedings of the 5th conference on Information Technology Education, 2004, pp. 6-12.

[6] D. Edelson, R. Pea, and L. Gomez, "The Collaboratory Notebook", *Commun. ACM*, vol. 39, pp. 32-33, April 1996.

[7] J. Lave, "Teaching, as learning, in practice", *Mind, Cult. Activity*, vol. 3, no. 3, pp. 149-64, 1996.

[8] D. Holt, D. Mackay, and R. Smith, "Developing Professional Expertise in the Knowledge Economy: Integrating Industry-Based Learning with the Academic Curriculum in the Field of Information Technology", *Asia-Pac. J. Coop. Edu.*, vol. 5, no. 2, pp. 1-11, 2004. [Online] Available: www.apjce.org/volume_5/volume_5_2_1_11.pdf. [Accessed July 21, 2008].

[9] L. Williams, and R Kessler, "Experimenting with industry's "pair-programming" model in the Computer Science classroom", *J. Comput. Sci. Educ.*, vol. 11, pp. 7-20, March 2001.

[10] M. Tanniru, and R. Agarwal, "Applied technology in business program: an industry-academia partnership to support student learning", *E-Serv. J.*, vol. 1, pp. 5, Winter 2002.

[11] B. A. Schuldt, "Real-world' versus `simulated' projects in database instruction", *J. Educ. Bus.*, vol. 67, pp. 35-39, Sep/Oct 1991.

[12] "What is service-learning?", *Corp. Natl. Commun. Serv.*, [Online] Available: www.learnandserve.org/about/service_learning/index.asp [Accessed July 21, 2008].

[13] R. Bringle, and J. Hatcher, "A service-learning curriculum for faculty", *Mich. J. Commun. Serv. Learn.*, vol. 2, pp. 112-122, 1995.

[14] T. Fox, "A case analysis of real-world systems development experiences of CIS students", *J. Inf. Sys. Educ.*, vol. 13, pp. 343-350, Winter 2002.

[15] M. Magboo, and V. Magboo, "Assignment of Real-World Projects: An Economical Method of Building Applications for a University and an Effective Way to Enhance Education of the Students", *J. Inf. Technol. Educ.*, vol. 2, pp. 29-39, 2003.

[16] A. Harris, "Developing the Systems Project Course", *J. Inf. Sys. Educ.*, vol. 6, pp. 192-193, 196-197, Winter 1994.

[17] 2006-2008 On-Line Graduate Catalog (2006) Winthrop University. [Online] Available: www.winthrop.edu/graduate%2Dstudies/csci.htm, [Accessed July 21, 2008].

[18] A. Porter, H. Siy, A. Mockus, and L. Votta, "Understanding the sources of variation in software inspections", *ACM Trans. Softw. Eng. Methodol.*, vol. 7, pp. 41-79, Jan. 1998.

[19] W. Humphrey, "Software – A performing science?", *Ann. Softw. Eng.*, vol. 10, pp. 261-271, Nov. 2000.

[20] S. Pflegger, L. Hatton, and C. Howel, *Solid Softw.*, Upper Saddle River, NJ: Prentiss Hall, 2002, pp. 194-209.

[21] R. Dromey, "Software Quality – Prevention versus Cure?", *Softw. Qual. J.*, Vol. 11, pp. 197-210, July 2003.

[22] T. Hoffman, "Preparing generation Z: CIOs say college graduates aren't ready for corporate IT jobs. Now some progressive universities are doing something about it. (Careers)", *Computerworld*, vol. 37, pp. 41-42, Aug. 2003.

Received: June 02, 2008 Revised: July 23, 2008 Accepted: July 24, 2008