A Company-based Framework for a Software Engineering Course

Thomas P. Way
Department of Computing Sciences
Villanova University
800 Lancaster Avenue
Villanova, Pennsylvania 19085
(610) 519-5033
thomas.way@villanova.edu

ABSTRACT
The subject matter of a typical undergraduate software engineering course, while providing necessary background, can be quite dry. Team-based programming projects often complement the more theoretical textbook and lecture content by giving students valuable hands-on practice, albeit on a small scale and within a traditional classroom setting. This paper describes a company-based framework used in two semesters of a software engineering course. This approach incorporates a novel, collaborative framework to simulate the real-world experience of working for a medium-sized software design company or research laboratory, while giving students a vested interest in the overall outcome.

Categories and Subject Descriptors
D.2.9 [**Software Engineering**]: MANAGEMENT – *programming teams*. K.3.2 [**Computers and Education**]: COMPUTER AND INFORMATION SCIENCE EDUCATION – *Computer science education*.

General Terms
Management, Design, Human Factors.

Keywords
Software engineering education, distributed group working, collaborative learning, team-based projects, pedagogy, capstone.

1. INTRODUCTION
Providing a grounding in theoretical concepts while reinforcing their importance in a real-world context is a major challenge in designing and implementing an undergraduate software engineering course [1]. Team-based programming projects frequently are used to engage students by affording them the opportunity to participate in the type of teamwork found in a software industry setting [2,6,8]. The use of such a team- or group-based approach, with small groups of three to five members each, has gained significant acceptance and support as a means to simulate the industry experience for software engineering students while reinforcing other course material [5].

The use of a larger project team format which more closely resembles the reality of the industry workplace is perceived as being more difficult to implement, and is infrequently reported in the literature [8,10]. When such an immersive model is used, however, students can learn more about "real" software engineering than with less ambitious frameworks [3,7,8]. Additional benefit is gained when students experience a distributed group working environment, where members of teams are physically, or virtually, separated while participating in cooperative tasks [4]. As the business structure of the software industry continues to become more distributed geographically, providing students with exposure to this way of working is a desirable part of their formal software engineering education and a valuable asset to their future employers.

This paper describes the design and use of two variations of such an industry-focused large group project as part of an upper-level undergraduate course in software engineering. Based on the author's extensive prior research laboratory and industry experience, novel attributes not normally found in a software engineering course were incorporated including class-wide product brainstorming sessions, overlapping subgroups of students, distributed group working, weekly engineering meetings, and business and marketing strategic planning aimed at releasing the finished product to the outside world. Observations, feedback from industry contacts, and the results of a follow-up survey with students are presented and discussed.

2. THE COMPANY-BASED FRAMEWORK
Two approaches to a company-based framework have been developed and used in the design of an undergraduate software engineering course. The company-based framework involves organizing students into a simulated software development company, with each student being delegated specific individual and group tasks plus a shared responsibility to the company for the design, development and ultimate distribution of a new software product.

The first variation models a university-based research laboratory and was used in a single section of the course. Based on experience with the first model, adjustments were made the

SIGCSE'05, February 23–27, 2005, St. Louis, Missouri, USA.

following year to produce a second variation that models a medium-sized bicoastal software development firm combining two sections of the course into a single entity. This framework attempts to simulate as realistically as possible the experience of programming-in-the-large as part of a large software development team developing a new product under deadline pressure.

2.1 Motivation and General Framework

"*Give a man a fish and you feed him for a day. Teach him how to fish and you feed him for a lifetime*." Lao Tzu

The principal motivation behind the use of a large group project framework was the belief that providing students with practical experience in working in a real software lab or company setting would reinforce the lecture and reading content of the course while demystifying the process of developing software in the real world. The initial vision was that of a newly hired computer science graduate sitting in a first engineering meeting and feeling completely at home, on familiar turf, ready to immediately contribute. To realize this vision required gaining complete buy-in by the students involved.

Engaging students in the process was accomplished by providing each a direct and personal stake in the outcome, beyond mere grades which were downplayed to the extent that was possible. A variety of justifications were presented, including experiencing exactly what a first job will be like, being a part of a project that will be released to the public at the end of the semester, having access to a permanent web product site to serve as concrete proof to potential employers of just what the student is capable, the chance to have a lot of fun and learn new technology, and the potential that the initial version of the product could lead to bigger things. Gaining the students' enthusiastic commitment amounted to being equal parts motivation speaker, experienced industry pro, and, occasionally, a red-pen-wielding professor.

The general framework for the course included a grounding in fundamental software engineering theory, including software processes, requirements engineering, system modeling and architecture, prototyping, user-interface design, verification and validation, project management and planning, software re-engineering and configuration management, all part and parcel of an undergraduate software engineering course [1,8]. Assigned readings and lectures were used to formally present this theoretical side, with lecture and discussion filling approximately 1.5 to 2.0 of the 2.5 hours of weekly class meeting time. Lectures were based on a required text [8], but were amply counterpointed with appropriate software industry anecdotes from the instructor's past. The flow of lecture topics was matched to corresponding stages of the hands-on product development cycle whenever possible. Midterm and final exams provided more traditional, and expected, assessments of the theoretical material covered.

As a result of a variety of individually produced student writing assignments throughout the semester, the course fulfilled a university requirement as a "writing intensive" course. All students crafted a rough design outline and prototype design, which led to an initial software requirements specification (SRS). Students were free to be creative and incorporate any new, unique and innovative features into their SRS that they could devise. A very complete final SRS served as a culmination of each student's individual design experience.

The remaining 30 to 60 minutes of class time was allocated to a weekly "engineering meeting." This meeting was run from an agenda provided to students in advance, with each student expected to be prepared to field questions that may arise within their area of responsibility at any time. Minutes of each meeting were recorded and posted on a continuously updated course web site, along with all lecture notes, assignments, deadlines and handouts. During initial weekly engineering meetings, the company was organized into smaller, overlapping teams, which formed the core of the two variations of the company-based framework. Later meetings addressed ongoing product design and development concerns, including product naming, feature identification, specification, problem solving, student demonstrations of web site and application versions, and other technological, business and aesthetic issues.

2.2 Novasoft Research Laboratories

In the Spring 2003 semester, one section of the software engineering course was organized into a simulated university-style research lab called "Novasoft Research Laboratories," or "NRL." On the first day of class, students were "hired" by NRL, and were presented with the main company goal: the production of an experimental image processing tool for release to the academic research community, and the world at large, by the end of the semester. The tool had to be designed for ease of use and extensibility, written in Java for wide compatibility, distributed as open source, and suited to legitimate research tasks. Thus, the "employees" were provided with a worthwhile, realizable and significant goal and a personal stake in the outcome.

The company structure is illustrated in Figure 1. The instructor served as Project Manager, and students were placed on the five teams based on interest or need. A very high degree of collaboration among teams was necessary and expected.

The **GUI Team** was responsible for development of the application user-interface, or front-end, which included an initial prototype, and for collaboration with the Module, Specification and Testing/Docs Teams. The GUI Team took the lead on integration of all modules into the finished product, known as the Villanova Image Processing Research (VIPeR) Tool (Figure 2).

The **Module Team** designed the software interface between the front-end and a generic image processing module, working very closely with the GUI Team. Later, this module interface was used company-wide to develop a range of plugable modules, resulting in information that was then provided to the Specification and Testing/Docs Teams.

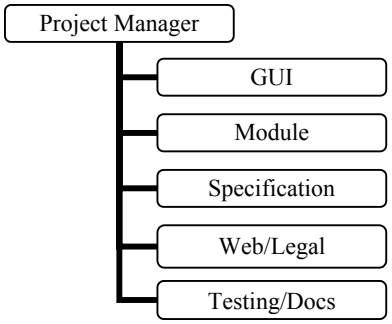

Figure 1. Organization of NRL Teams

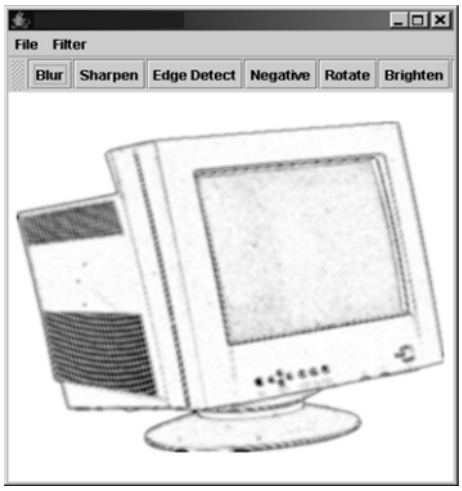

Figure 2. VIPeR Tool, edge detection filter.

The **Specification Team** had the job of gathering the best of individual ideas and features, and incorporating those ideas in a shared final product specification. Design information and product screen shots were gathered from the GUI and Module Teams for use in the specification document, and also passed on the Testing/Docs Team.

The **Web/Legal Team** was charged with developing a company web site [11], including general product description, user support, a legal licensing agreement (a variation of the Gnu Public License), and product download capability. The Specification and Testing/Docs Teams provided content to the Web/Legal Team for inclusion on the site, while hands-on experience with early versions of the product in collaboration with the GUI and Module Teams was needed to generate user support information.

The **Testing/Docs Team** was responsible for devising a testing plan and working with the GUI and Module Teams to test and debug the product. It also collaborated with the Specification Team to write user and developer documentation, and created a product installer that was given to the Web/Legal Team to enable easy distribution of the product online.

Because of the relative autonomy of teams, twice during the semester team members evaluated each other and themselves by assigning a "contribution factor" of 0 to 100 to each member of the team along with a brief justification of the score. These evaluations were used primarily to identify and rectify "weak links" in teams, assuring that members fully contributed to the team effort. The unavoidability of weekly public-accountability during the engineering meeting seemed to serve as additional motivation for participants to get things done. It was the rare student who failed to contribute sufficiently to the effort.

2.3 Novasoft Game Laboratories

Based on the experience with NRL, the course for the Spring 2004 semester was modified to incorporate multiple, overlapping subgroups and a project with more student appeal. Students from two sections of the course were organized into the "Novasoft Game Laboratories," or "NGL," and given the mission of designing a game-playing software product with three goals in mind. First, it must be a fun-to-play game system that allows a user to enjoy a variety of games, suited to their taste. Second, it must be an extensible platform that allows other programmers or students to create and integrate their own game modules into the product. Third, it becomes a calling card for each team member to use in the future to illustrate their talents, providing experience in an actual software development group.

Organization of the company was more complex. First, members of each section were assigned to one of two geographic office locations: section 1 was at "HQ" located in Palo Alto, California, while section 2 was the "East" office located in Nashua, New Hampshire. The virtual geography of these locations were to be respected, with collaboration between members of different sections of the course to be done either electronically or, with permission of the Project Manager, face-to-face having been approved for such "travel" to the other location.

Students were each assigned to two distinct teams: a **game module development team** and a corporate-level **product development team**. Game module teams consisted of three to five members from the same "office" (section), and were responsible for the complete development from design through implementation of a computer game that integrates seamlessly into the overall product. It was left to the discretion of each team what specific game was developed, although implementation as either a Java application or applet was mandatory.

Product development teams were organized in a similar fashion to those in NRL (Figure 3). Some of the responsibilities of the earlier incarnation of this company-based approach were redistributed to achieve a better balance among workloads for teams. Significant interaction between teams again took place.

The **Front End & Integration Team** was responsible for designing an initial prototype, and for creating the framework to make game module integration work. The resulting application was **JavaGP**, a Java Game Playing application (Figure 4). For the integration phase, company-wide collaboration with all individual game module teams was performed.

The **Web & Legal Team** created a company web site which included product description, product and individual game module download capability, and a password-protected member contact database to facilitate the high communication rate among students at the game module team, corporate development team and company-wide level [11].

The **Specification & Documentation Team** again elicited the best ideas from all students, merging them into a complete product software requirements specification and associated developer and user documentation.

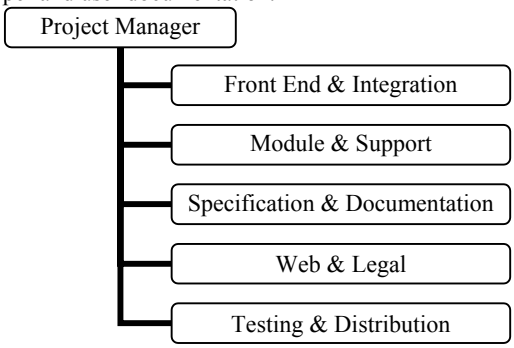

Figure 3. Organization of NGL Teams

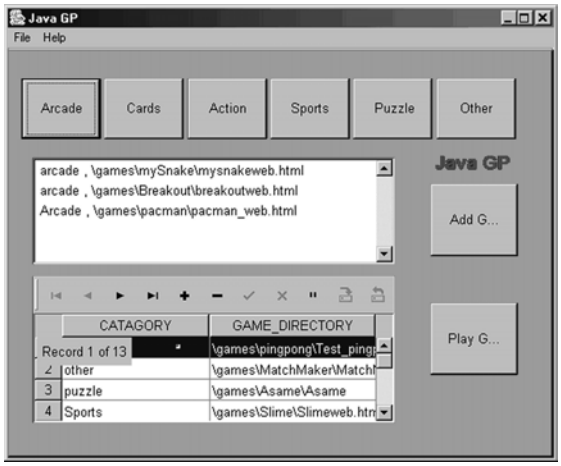

Figure 4. JavaGP main user interface screen.

The **Module & Support Team** collaborated with the Front End & Integration Team to design a flexible interface for adding and playing Java application and applet games via the user interface. This approach was disseminated to all game module development teams to guide their design efforts. A text-based configuration file provided a robust module information database, enabling the addition, deletion and modification of game-related display and execution information to be manipulated via a parser within the user interface or directly using a text editor.

The **Testing & Distribution Team** joined forces with the Front End & Integration Team early to develop and test the product as it evolved from an initial prototype to the completed product. Significant effort was expended creating and testing the executable installer version of the product, since the final version included the Java runtime environment, a myriad of application and applet class files, and various data and image files.

Team member evaluations were again successfully used to assess individual contributions, identify problem areas, and to correct lapses in participation, which were rare.

2.4 Evaluation of Product Outcomes

A single semester is a very short time for a complete software product development cycle, and as a result the software produced is not in a polished state. The evolutionary Unified Process Model used was a good fit given the time constraints, enabling ongoing integration of incremental improvements to the product. The image processing tool still has many bugs related to the user-interface (mostly issues with event-related painting and Java), and fewer interesting image processing modules than a DSP investigator would require, but the framework for adding more modules is well designed and the tool potentially could be a good starting point for a researcher.

The game software is further developed, thanks to a small group of students who took the initiative to work many late nights, solving problems and integrating individual game modules produced by their fellow students. The flexibility with which new modules can be added is solid, and the design decision to use a text file to use the initialization information makes the possibility of downloading and installing upgrades a very simple matter. The use of Java provides very good cross-platform compatibility, and the program seamlessly allows running of games implemented both as applications and applets.

The finished products were solid efforts, especially for unseasoned upper-level undergraduate students. The results were far from professional quality, but that is to be expected given the limited time available and the backgrounds of those involved. The more important result is that the collective group of 25 to 45 students collaborated to accomplish a significant outcome under a strict deadline and with little prior experience. The instructor served as the overseeing Project Manager, but it was the students who poured their creative energy, programming skills, and many long hours into development of the products.

3. OBSERVATIONS AND ANALYSIS

In general, the company-based framework of the courses was successful in exposing students to the theories and practical aspects of software engineering. Students clearly felt engaged in the process, and their enthusiasm fed off of each other. When weekly meetings would, as engineering meetings tend to, digress into the minutia of web site logos or fonts used on user interface buttons, a fair amount of latitude was allowed by the instructor, since recognizing the tendency to digress is a valuable survival skill for a software engineer. The company frequently contracted a nasty case of "creeping featuritis," the tendency to continuously brainstorm new product feature ideas to the exclusion of getting any real work done. This discussion was included in the design process, and students were encouraged to learn to recognize and address these very natural inclinations.

3.1 Survey and Observations

Students were surveyed and asked to provide their assessment of the value of the course as it was structured to both their academic and professional career. An overwhelming majority of students stated that the course was very or extremely valuable on both fronts. Many students have positively drawn on their experience in the software engineering course while undertaking another significant and more-autonomous project as part of the capstone Senior Projects course.

Many students have graduated or held summer jobs in a variety of positions involving computer game development, aerospace, financial, insurance and software industries. All have expressed enthusiastic appreciation for the company-based approach. A sample of the typical remarks from former students includes:

- "I probably had the greatest job ever this summer, making the Madden 2005 video game. I was shocked and most pleased I got to use what I was actually learning in school."
- "I worked for Raytheon this summer and applied all of the software engineering principles we learned. It has been the most valuable class that I have taken at Villanova."
- "I specifically remember being able to use my experience from the course as a response to an interview question with Vanguard. It must have worked, because I got the job and I'm with the company!."
- "This summer I worked at Siemens Medical writing test scripts for their new software. It was great that I could go into the job knowing terms, procedures, and methods of testing software. Everything fell into place when I experienced it with a large software project."

- "I worked in several groups over the summer in my internship and I really used the skills I learned because we had people who did work, people who did not, and people that tried to take over."

Some students were concerned that the self-supervised nature of team tasks was detrimental, due to the natural tendency of students to put off those items which do not have a fixed deadline. A few remarked that the sheer volume of theoretical material covered was excessive, while others noted that imbalance was unavoidable in assigning and carrying out various project tasks.

Recommendations included using more industry-standard tools for modeling and version control, requiring all students to participate in more hands-on programming, and tying company activities more completely to all lecture topics. One student noted that smaller group projects would guarantee that students could not "hide within their comfort zone" since they would be responsible for all phases of product development. Many recognized the value of the company-based framework as being directly applicable to their current or future positions in industry and academia. Additional feedback from students is expected with each iteration of a planned longitudinal study.

Because each company served as its own client, experience in client-interaction, such as requirements elicitation, presentations, and executive buy-in, was lacking. The skills gained in dealing with the instructor-as-client, however, offset some of this shortfall. In prior, small-group versions of this course, student teams sought out "clients" in various university offices to gain client contact experience.

3.2 Industry Feedback

A number of current and former colleagues now working as software engineers and project managers were informally surveyed for feedback on this company-based framework. Comments such as "I wish we had something like that when I was in school" and "Let me know when your students are graduating" were common. No reservations about lack of individual level of participation were expressed. Rather, the consensus was that having new hires already intimately familiar with the business model and normal functioning of a large project team were far more valuable than a more specific background in programming or testing, for example. Most said that they would expect a learning curve as far as the technology was concerned regardless of the level of experience of a new employee, so having that new employee feel comfortable within the team environment, ready to contribute to the process right away would be very desirable to them as a team leader or project manager.

4. CONCLUSIONS

The company-based framework reported in this paper attempted to provide students with a simulated real-world software engineering experience to reinforce the theoretical aspects of the subject. A strong case can be made for the effectiveness of this approach, with positive reviews from the instructor, students and interested industry representatives. A needed improvement is routinely achieving work-load parity among team members.

While there is clearly a place for smaller group projects in a computer science curriculum due to the necessity of hands-on experience with software design and implementation, software engineering as it is practiced in industry is frequently a large group activity. The software engineering course perhaps is the only opportunity within a general computer science curriculum to engage students in the experience of programming-in-the-large, with a company-based approach such as that reported in this paper presenting a viable and valuable option.

5. ACKNOWLEDGMENTS

Thanks to my CSC 4700 students from the Spring semesters of 2003 and 2004 for coming along for the ride, and to Dr. Dan Joyce and Dr. John Lewis for their valuable advice and feedback.

6. REFERENCES

[1] ACM Computing Curricula, October 2001. Available: http://www.acm.org/sigcse/cc2001/steelman/, 2001.

[2] Adams, E. J. *A Project-Intensive Software Design Course*. In *Proceedings of the 25th ACM SIGCSE Technical Symposium on Computer Science Education*, Indianapolis, Indiana, 112-116, March 1993.

[3] Boehm, B. W., Egyed, A., Port, D., Shah, A., Kwan, J., and Madachy, R. J. *A Stakeholder Win-Win Approach to Software Engineering Education*. Ann. Software Eng. 6, 1295-321, 1998

[4] Brereton, P., Lees, S., Gumbley, M., Boldyreff, C., Drummond, S., Layzell, P., Macaulay, L., and Young, R. *Distributed Group Working in Software Engineering Education*. Journal of Information and Software Technology (IST), special issue on Software Engineering Education, 40, 221-227, May 1998

[5] Gotterbarn, D. and Riser, R. *Real-world Software Engineering: A Spiral Approach to a Project-Oriented Course*. In *Proceedings of the 7th IEEE Conference on Software Engineering Education and Training*, 119-150, 1994

[6] Habra, N. and Dubois, E. *Putting into Practice Advanced Software Engineering Techniques through Students Project*. In J.L. Diaz-Herrera (ed.), *Proceedings of the Seventh Conference on Software Engineering Education,* volume 750 of LNCS. Springer-Verlag, San Antonio - Texas, January 303-316, 1994

[7] McCauley, R. A., Adams, E. J., Gotterbarn, D. J., Northrop, L. M., Saiedian, H. and Zweben, S.. *Organizational issues in teaching project-oriented software engineering courses*. ACM SIGCSE Bulletin , Proceedings of the Twenty-Fifth SIGCSE Symposium on Computer Science Education, Volume 26, Issue 1, 392-393, 1994.

[8] Northrop, Linda M. Success with the Project-Intensive Model for an Undergraduate Software Engineering Course. SIGCSE Bulletin 21, 1, 151-155, February 1989.

[9] Somerville, I. *Software Engineering*. Addison-Wesley, 6th Edition, 1996.

[10] Tomayko, J. E. and Shaw, M. *Models for Undergraduate Project Courses in Software Engineering*. In J. Tomayko (ed.) Software Engineering Education (*Proceedings of the 5th Conference on Software Engineering Education)*. Springer-Verlag, pp. 33-71, 1991.

[11] Way, T. P. Software Engineering course web sites, Villanova University, http://www.csc.villanova.edu/~tway/, 2004.