FINDING AN "IDEAL" MODEL FOR OUR CAPSTONE EXPERIENCE*

Nan Sun, Jack Decker
Computer Information Sciences
Washburn University
1700 SW College Avenue
Topeka, KS 66621

ABSTRACT

The Computer Information Sciences (CIS) department at our University is in the process of charting the future direction of its required capstone experience. This process involves evaluating the experience the department has had with a one semester project oriented class and looking at capstone experiences at other universities. The purpose of this study is to identify an ideal model or models for our capstone experience that would be most beneficial for our students while fitting into the mission of the University. Our research is leading us toward a two-option experience, one for those students destined for graduate school and another for those who will seek employment immediately after graduation. Proposed for the former is a research class and for the latter the establishment of a faculty directed but student staffed consulting group with the mission of supporting non-profit social service organizations. Anticipated issues and challenges of both options are detailed.

BACKGROUND

Since 1994, the Computer Information Science department at our University has required majors to participate in a Capstone project. The goal of the required Capstone course is to provide closure for Computer Information Science majors. The intent is to allow students an opportunity to assimilate and synthesize the knowledge acquired during their course of study for the major in a group project setting. The one semester course is taken during the senior year, is for 2 credit hours, and is offered for credit or non-credit (i.e. letter grades are not assigned).

Projects selected for the Capstone classes have varied over the years and several different faculty have been assigned to the course. Some of the assignments have been strictly classroom projects while other projects have been for local businesses. Priority has been given to projects favored by students in the course and to those requested by friends of the University. Most, but not all projects of late have been Web applications.

Regardless of how a project is selected, the intent is to make one pass through the development life cycle. Deliverables include: project description, project plan, requirements and design specifications, data primitives such as records of hours worked on various tasks, code, and user manual. In all cases (including those for real customers), the final project is delivered without warranty and at best is considered a prototype.

The general feeling among current faculty is that the Capstone experience needs to be upgraded, although there are certain positive aspects to the current methodology that should be retained, including the requirement for group work. Experiencing the highs and lows of depending upon critical work flow from a group member is valuable preparation for work environments. It is also a benefit to students to see the development process from start to finish on a project that is more challenging than typical class projects. Providing a situation that requires students to integrate and build upon previous work and learning is also important and should be retained. Finally, such an experience gives students something concrete to discuss with recruiters and to put on a resume. Even though not "ideal", the current capstone experience has merits.

On the other hand, some aspects are in desperate need of attention. Even though there are advantages to having students work in groups, there are pitfalls as well. Current practices have not been very successful in evaluating the contribution of each group member. Students have been judged on group output, not on individual output. Leaders emerge in the groups and sometimes these leaders elect to do most of the work instead of depending on perceived less capable or less reliable group members. Furthermore, since grades are either pass or fail, faculty really cannot use the grade club to urge better performance. Students have quickly learned that if they do something and present it in a reasonably coherent manner, then they and the rest of their group will pass. Students also do not expect to have to do too much work to pass a 2-credit hour course. Faculty also have the feeling that the processes used in the capstone needs to be more clearly defined and more consistently applied. Such processes include those to solicit projects, manage projects, bring the quality of the products to production levels, and to provide maintenance for products to be installed by real clients. Finally, some students resent that an applied project course is required when the student's intent is to immediately attend graduate school after graduation. The feeling is that a research course would be more beneficial in these cases.

IDENTIFYING AN "IDEAL" CAPSTONE MODEL

A variety of approaches have been taken by CIS and Computer Science (CS) departments at other universities to provide a capstone experience. After looking at online catalogs and reviewing the literature it appears the following models are common.

Course Based

In this model, students are required to take either a fixed list of courses, choose an emphasis and take the required courses for that emphasis, or simply take a specified number of hours of elective courses. The University of Wisconsin Eau Claire's approach to the capstone fits into this category. Majors are required to take 6 credit hours in one of two tracks. The research track required the completion of Research in Computer Science I and Research in Computer Science II. The other track requires the student to take Software Engineering II and either Computer Science Practicum or Computer Science Internship. The Practicum course requires a comprehensive project and the Internship practical work experience. [8]

Classroom Project

This approach simulates the business and industrial development environment, but does it entirely within the confines of the university without a real customer. At least one software development life cycle is attempted during a semester and the application of software engineering principles are stressed. Work is done in groups with a goal of motivating students to be better group members and to improve their communication skills. The delivered product is never the primary objective, learning is. [2]

Client-Based Software Development

Students work on actual projects for real customers. The customers could be local businesses, industrial partners, and units within the University. As with the other models, students apply software engineering techniques, work in groups, write software, test software, and prepare user manuals. The primary difference is that this is done for the benefit of a real customer. A good example in this category is the University of Nebraska at Kearney. Their web site contains an assessment of their capstone project experience from 1993 to 2001. [7]

All of the above approaches have merits and, if properly executed, would certainly provide a meaningful capstone experience for students majoring in CIS or CS. While doing our research, however, we found a paper [1] that greatly influenced our thinking. The paper advances the benefits of using "**socially relevant**" projects for the capstone experience. The authors found that student's using more traditional approaches "demonstrated a noticeable lack of attachment and interest when working on projects lacking applicability outside of the classroom." They also note that "industry partners are not likely to turn over large, complicated, important project to temporary student help." Furthermore, they observe that the capstone often turns into another game students play to complete their degree and a high grade becomes the primary goal.

The approach advocated by the authors of the referenced paper is to select local customers **who are in need**. Potential customers might be the United Way, the Salvation Army, and/or health care organizations. As with other capstone models, students are assigned the task of analyzing, designing, and implementing software in a group environment. And while the primary goal is still student learning, an additional benefit is that students provide useful products for organizations that could not otherwise afford

them and students come to see their work as important and worthwhile. It's no longer "just" a classroom exercise, it's "helping someone else." The natural consequence of this approach is that students are more motivated, they produce higher quality work, and they have a greater sense of accomplishment.

Having considered the merits and drawbacks of our current approach to the capstone experience and the capstone experiences of other universities, and seeing much merit in the socially-relevant approach, it becomes apparent that the "ideal" capstone should:

- Challenge students to demonstrate their proficiency in CS or CIS
- Require the use of sound software engineering methods
- Require that students work in groups
- Motivate students through client-oriented projects that students feel are important and relevant
- Produce high quality products that once implemented, can be successfully used by clients and their customers
- Provide consistently positive experiences for students from semester to semester
- Allow students who intend to go directly to graduate school to concentrate on research projects for their capstone experience.

THE DUAL-OPTION MODEL

Our research and experience has led us to a dual-option model:

Option One is to provide a research class (RC) for students wishing to go directly to graduate school. The goal of the RC option is for a student to specialize in an area, then write and submit an article for conference presentation and/or publication. This option would require extensive reading and research on a focused area under faculty guidance.

Option Two would be a client-based socially relevant project (SRP). Students would work together in groups of 4 or 5 on projects for non-profit social service organizations such as United Way and Salvation Army. The intent would be to motivate students through means other than grades to produce production ready applications and to help students mature through community service. This option would be for those wishing to pursue a software development career after graduation.

To fulfill degree requirements, a student would need to choose one of the above options.

Benefits

The benefits of the RC option are two-fold: one-on-one supervision provides better chances for students to excel in their research projects and leads to improved faculty-student relationships. Assessment for this option is relatively straightforward. Whether the paper is accepted and the level of the conference it is accepted by can be direct measurement of the student's performance.

There are abundant benefits in the SRP option. It is relatively easy to establish partnerships with non-profit social organizations. Unlike industrial partners, non-profit

social organizations, due to their limited resources, will be more likely to welcome free services and be more appreciative of computer technology they could not otherwise have had access to. With the aid of the applications provided, the organizations, in turn, will provide better and more efficient services to the community. Successful projects will improve university-community relationships as well.

Students will certainly benefit as well. What they experience in the process will be invaluable and is exactly what practitioners will face outside of the academic sphere. Students will be required to work with non-technical clients who often do not understand technology and do not use the same vocabulary to describe system features and constraints. This will temper the urge to jump into coding, which many students cannot resist doing. They will be forced to devote time to understand the issues, analyze requirements, and to carefully design the system.

In the SRP model students will be motivated to do their best because they know that a valuable service will be provided by the end product. By visiting the clients and understanding what they do, students will be exposed to environments which will help them develop a sense of community and better relate what is learned in the classroom to its applications in society. Part of the mission of our University is to provide positive transformational experiences for students during their college career. The capstone experience will help meet this goal. Other obvious advantages for students are that such work can be included on their resumes and that the experience provides the opportunity to do valuable networking with employers and organizations for future job possibilities [4].

Challenges

The RC class has drawbacks. One is: since the paper focuses narrowly on an area, this approach cannot serve as an assessment tool for the overall degree program. Secondly, if not carefully guided, students may choose this option simply because they prefer to work alone and avoid the complications of working with other students.

The SRP option also faces major challenges. Since this option is far more complex, these challenges are grouped in the following categories: quality assurance, maintenance, faculty and student load, and ownership and liability.

Quality Assurance: When software development is just a classroom exercise, it is ok for the product to not be fully implemented and tested as long as the students are learning in the process. However, if the software is to be used by real clients, then it must meet all requirements and be reasonably error free before it is put into use. In other words, quality becomes priority number one. Currently, quality assurance topics, along with many others, are briefly covered in a senior level introductory course in software engineering with minimum practice time. Integrating these theories into a capstone project will constitute an excellent learning opportunity, and the students surely will be much better prepared when they move on to other projects. But as far as the capstone goes, their limited experience may mean the quality of the product cannot be assured. This potential quality problem raises a fundamental question: how should we prepare the students from the very beginning of the program for a successful capstone experience?

And in particular, how early in the curriculum should software engineering techniques be introduced?

There are also situations where the best results are achieved when a group outside of the development team is involved. For example it has been proven that testing is more thorough and objective if not done by people on the development team. Thus, what group and class structure is ideal for achieving quality goals?

Maintenance: The delivery of the product to the client marks the end of the development phase. But maintenance usually starts right away, and it is far more than just fixing errors. As a matter of fact, there are four types of maintenance: corrective maintenance, adaptive maintenance, perfective maintenance or enhancement, and preventive maintenance or reengineering. [6]

This brings up two issues. Issue one has to do with who should maintain the product. There are three possible avenues: the development group, a new capstone group, or other students. All three options face problems: The development group is not a good option unless the capstone project is a yearlong experience. This group is expected to move on to other classes, and some members may have graduated when maintenance starts. If the maintenance task is assigned to a new capstone group, the unpredictable nature of maintenance work will make it difficult to ensure these students will carry a load similar to groups doing development. Also, no student should miss the opportunity to experience the full software development life cycle in a capstone project. The last option is to assign the task to other students. In this case, the challenge is to anticipate and satisfy the ad hoc and unpredictable work needs through hired student labor or through internships. If hired, who would fund the work? If internships, how would such experiences fit into the overall curriculum?

The second maintenance issue is that the maintenance schedule often does not match the academic calendar. The client's business usually runs all year around. Therefore, at least corrective maintenance should be performed whenever the need arises. How can this be accomplished when faculty and students are not available, for example, during summer and winter breaks?

Faculty And Student Load: Not all of the work required to successfully implement the SRP model can be easily represented by a fixed number of teaching hours. Identifying suitable organizations and projects, and maintaining partnerships can be time consuming to faculty, and the amount of time devoted will vary from semester to semester. Furthermore, some, if not the majority, of support and maintenance work, will be requested spontaneously. Consequently, it will be a real challenge to fit the SRP work load into an education system where students earn a fixed number of credit hours for graduation and faculty teach a fixed credit hour load every semester.

Liability And Ownership: Ownership of the product can become an issue, especially when the product reaches a level to have potential market value. So, who owns the product, the student group, the university, or the client? It is commonly understood that when someone uses company time and resources and is paid by the company to develop a product, the company owns the product. In the SRP case, the issues are less clear. From the client's point of view, they may not want to use a product that is not client owned. On the other hand, the University may claim ownership since the University pays

faculty to supervise students who are participating in a university class. Or maybe students own the product. After all it is students who analyze, design and implement the project.

Liability is another touchy issue. Since nobody knows how to make zero-defect software [3], liability is inevitable. The Volunteer Protection Act (VPA), signed into law by President Clinton on June 18, 1997, generally permits volunteers to serve without fear of liablility, but the VPA's limitation of liability does not extend to the organization itself [5]. So when the inevitable software error occurs that results in revenue loss, who is responsible and can the University's liability be limited?

PROPOSED IMPLEMENTATION STRATEGIES TO ADDRESS CHALLENGES RAISED IN THE DUAL-OPTION MODEL

It should be clear at this point that the implementation of an effective Capstone experience is not easy and that implementing a SRP option is a particular challenge. It is not at all clear that our University will decide it is in its best interests to implement such a model. It may elect to stay with a more traditional model for the capstone experience. In any event, the authors have considered and will now present their view of how the Dual-option Model could be implemented.

To ensure that graduates from our program have good understanding of various subject areas, all students, (whether selecting the RC or SRP option) should be required to pass a comprehensive exam that covers topics in data communication, computer architecture, object-oriented programming, operating systems, and software engineering. This addresses the issue that students may only narrowly focus on one specific area.

Students who are interested in the RC option would be required to submit a written proposal to the faculty capstone research committee one semester before the research class starts. The proposal would include research topic, methodology, and the conference/journal the paper would be submitted to. The committee would review the proposal and decide whether to accept or deny the proposal. If the proposal is accepted, the student would enroll in the research class and would be assigned a faculty advisor for the paper. If the research proposal is denied, the student would be allowed to resubmit the proposal after revision or choose the SRP option.

The implementation approach for the SRP model should be to establish a faculty-directed and student-staffed consulting group. This consulting group would utilize students as programmers and analysts and faculty as project managers or consultants. Faculty and students would jointly solicit projects, determine project priority and feasibility, and assign groups to projects. Each project would start with contract negotiation. The contract would address all issues related to the project, in particular, scope, deadline, client and developer's roles in quality control, maintenance, liability, and ownership. During the contract negotiation phase, the consulting group would confer with university counsel whenever necessary to avoid potential legal issues. No project would proceed without mutual agreements in these areas. To guarantee a quality product, students and clients would work jointly on the project to identify and agree on areas where high quality must be achieved, and develop a test plan to thoroughly test those areas before the product is put into use.

Instead of trying to fit a project into a semester, contracts should be negotiated with clients for durations appropriate for the project, not to follow the academic calendar. Student involvement would be for the entire duration of the contract, although particular students would enroll one semester at a time. One possibility is for students to take the software engineering class to gain theoretical background and then enroll as interns during their junior year before taking the Capstone class in their senior year. Juniors would primarily serve as programmers and senior capstone students as analysts and/or lead programmers. Students would also be encouraged to join the consulting group for credit or simply for experience. At any given time, multiple projects could be active and students could take roles in more than one. Continuity from semester to semester would be provided by returning students and by faculty assigned to the consulting group. Students and faculty would be expected to provide support during breaks in the academic calendar. Student and faculty commitment would be essential to the success of this consulting group.

The University and the department should support this effort by adjusting credit hours for students and reassigning at least a portion of the teaching loads of two or more faculty members to internal consulting group supporting the SRP capstone option. We also believe that the capstone project should be at least three credit hours for students, ideally six credit hours for the SRP option. The grading policy for both options should be a letter grade, not Pass/Fail. Individual students should be evaluated, not groups. Client feedback, bi-weekly peer evaluation and software tools such as version control could be used to evaluate student performance.

CONCLUSION

A dual-option model is proposed for the "ideal" capstone project: a research class (RC) and a client-based socially relevant project (SRP). The goal of the model is to provide a learning experience that is both appropriate for the long term goals of CIS and CS students and provide an environment that highly motivates our students to do quality work. By being involved in socially relevant projects, the hope is to promote good citizenship along with good software development skills.

The authors are also well aware that there are significant challenges associated with this model. Nonetheless, we believe the benefits of the model would be numerous and the successful implementation of this model is feasible. What it requires is the resolve and ongoing commitment of the Department, the University, and individual faculty. Good students are also a must.

EPILOGUE

As of this writing (June, 2004), negotiations are in process between the University and a potential client to define the department's first capstone project fitting the SRP model. The proposed project will span three semesters and provide a very much needed software product for the client organization. If negotiations are successful, more details about this project will be provided during the conference presentation of this paper.

REFERENCES

[1] Buckley, M., et al, (2004), *Benefits of using socially-relevant projects in computer science and engineering education*. SIGCSE'04.

[2] Conn, R. (2004) *A reusable, academic-strength, metrics-based software engineering process for capstone courses and projects*. http://www.sheridanprinting.com/sigcse04/files/p48-conn1.pdf, retrieved March 2004.

[3] Kaner, C. (1997). *Software liability*. http://www.badsoftware.com/theories.htm, retrieved March 2004.

[4] McCoy, R., and Wymer, S. (2004). *An information systems project management course using a clint-based model*. Proceedings of the 7th Annual Conference of the Southern Association for Information Systems.

[5] Milestone, J. *Tort claims immunity for healthcare volunteers*. http://www.peick-usa.com/forms/Milestone%20re%20healthcare%20volunteers.pdf, retrieved March 2004.

[6] Pressman, R. (2001). *Software Engineering a Practitioner's Approach* 5th edition. McGraw Hill.

[7] University of Nebraska at Kearney, http://aaunk.unk.edu/asmt/oldDeptAsmt/csis/csiscc.htm, retrieved March 2004.

[8] University of Wisconsin Eau Claire on-line catalog, http://www.uwec.edu, retrieved March 2004.