INCORPORATING REALISTIC TEAMWORK INTO A SMALL COLLEGE SOFTWARE ENGINEERING CURRICULUM[1]

Ellen L. Walker, Oberta A. Slotterbeck

Computer Science Department

Hiram College

Hiram, OH 44234

{walkerel, obie}@hiram.edu

ABSTRACT

Software Engineering is by nature a collaborative process. Nearly all production software is designed by teams, working over a long period of time, using software tools to aid in the management of large, complex projects. In the real world, software projects continue for years, team composition changes, and a legacy library containing both documents and code provides project continuity. Unlike most teams created in a traditional classroom, typical teams have diverse backgrounds and strengths. This paper describes our efforts in building more realistic teams within a small college with traditional and non-traditional students. We describe how we have incorporated teamwork into our curriculum, as well as the challenges that teams face in our environment. Suggestions for addressing these challenges are proposed.

1 INTRODUCTION

Software engineering is the process of designing, building, and maintaining large software systems [9]. In industry, for each software product, many teams are involved in the software engineering process from initial design and specification through maintenance. Product development cycles last years, and individual software engineers move between products and take on new roles as they advance.

Teaching software engineering to undergraduates is a difficult task. The academic calendar assumes that each field can be broken into topics that can be learned, applied, and evaluated within a term of at most 15 weeks. Evaluation criteria emphasize the learning of students as individuals. These constraints are best satisfied by courses where students develop small, structured projects, individually or in small

[1] This material is based upon work supported by the National Science Foundation under Grant No. 9952749. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation (NSF).

teams, to clear specifications developed by the instructor. Unfortunately, students whose software development experience is limited to these courses are woefully unprepared for industry – they have the technical skills and scientific preparation, but not the necessary people skills and process skills to compete [3].

In engineering programs, many of these issues are addressed by a capstone sequence — two courses taken in the senior year where students from multiple departments work in teams to design a system to satisfy a customer's requirements. Similarly, many universities have added a senior capstone course or sequence to their software engineering programs, in which student teams develop a software product, often for a customer, producing appropriate milestone documents along the way. Like engineering capstone courses in general, these courses have a long sequence of prerequisites. In a small college environment, we cannot afford to require long sequences or concurrent courses. Our need is to achieve the educational objectives of a capstone course within the constraints of a small college program.

2 RELATED WORK

A well-known example of a capstone course is Georgia Tech's "Real World Lab." Projects in the laboratory have a six-month to five-year development timeframe, though students take the course for two to three quarters. As in a real company, as students become more experienced, they move up in their teams, and each semester new teammates are trained in the current product and process. Real customers provide projects and test data, and are directly involved in project review (requirements, documentation and beta testing).

Many programs have project courses that finish a yearlong sequence in software engineering [4, 6, 7]. Student teams must deliver appropriate artifacts from feasibility plans through implementations and final documentation. Often, students are assigned to formal roles within their teams [6, 10], and both individuals and teams are evaluated.

At Victoria University of Wellington [2] emphasis is placed on supporting teamwork in the capstone course. Students are given clear process guidelines, and assessed on both the outcome of their project and the effectiveness of their team. Along with the usual written documents, each student writes an essay on managerial lessons learned" which is made available to future students in the course. Highlights from these essays are incorporated into a tutorial on teams included in the course.

An alternative to a capstone sequence is a project shared over multiple concurrent courses. At California Polytechnic State San Luis Obispo [11], a single project is shared between a software engineering course and a database course. The project solves a real problem for a real customer. Deliverables include a database model, a relational schema, and an implementation for the database course; and a project plan, requirements document, and design document for the software engineering course.

In studying these examples, we draw three conclusions. First, **a single term is insufficient time to introduce the concepts of software engineering and also carry out a significant team-based project from requirements through delivery**. Programs either require a multi-course sequence or concurrent courses (with prerequisites!) to provide a significant software development experience. In our small-college environment, it is not possible to offer courses often enough to require

long sequences or concurrent courses. Therefore, we must develop a looser structure that can still provide multi-course experiences for individual students.

Second, **the process of software development is learned by developing software under conditions similar to those used currently in industry.** Therefore, students are grouped into teams of three to six, develop progress reports and milestone documents, and formally present their resulting solution. In industry, team members have diverse backgrounds; in engineering capstone courses, diversity is provided by combining students from different departments. When possible, software is developed for real customers [5, 8, 11], who approve requirements documents, prototypes or specifications, as well as the final product. Students are provided with production quality tools [2, 4, 5, 6]. In our curriculum, we strive to keep the process realistic, though it is difficult in short courses. We use real customers in the Database course, and we are teaching production-quality tools such as Oracle and Rational Rose. Diversity of background is provided by mixing students from multiple degree programs and different educational levels.

Third**, communication and teamwork skills, including writing and presentation, are a crucial part of a software engineering curriculum.** The importance of working together as a software development team is emphasized in course syllabi and evaluation criteria [2, 6, 8]. Students perform feasibility studies and requirements elicitation [4, 8, 11] to develop formal requirements and design documents worth a significant portion of their grade. In multi-course sequences, formal progress reviews [8] and web pages [2, 5, 6, 8] provide continuity. Similarly, all of our courses include formal oral presentations with supporting written materials.

3 CURRICULAR CHANGES

In our new software engineering curriculum, we are incorporating multi-semester, multi-course projects that require students to work together in teams. Each course has at least one significant team project, but not all projects are started from scratch. Instead, teams are given "software artifacts," works-in-progress from prior courses. Because students can enter the program with any of the courses, team composition changes from course to course, and teams include students with different backgrounds and levels of experience. The courses that are included in this curriculum are Database Design, Project Management, Interface Design, Systems Testing and Verification, Systems Analysis, and Software Evaluation. During the first year of the new curriculum, three of the courses were taught: Database Design, Project Management, and Interface Design. An additional course, Multimedia, was taught in a similar format with team projects. This paper reports on our experiences with diverse teamwork in these courses.

3.1 Team Composition

Students in our software engineering sequence come from two separate populations: weekend students, who have maturity and industry experience (though not always in the software industry), and traditional students who have more programming coursework, but do not see the value of the process aspect of software engineering. The weekend students are majoring in Computer Systems Management, which is an interdisciplinary major from the departments of computer science and management, while the traditional students are computer science majors. Teaming students from the two populations encourages them to learn from one another. We expect the weekend students to provide their management and process expertise to the

teams, while traditional students provide technical expertise. Therefore, in each course, we require that teams consist of a mix of traditional and weekend students.

In a small liberal arts program such as ours, it is not possible to offer every course every term. Therefore, students cannot progress in lockstep through a graduated sequence of courses. Instead, both the traditional and weekend curricula begin with a two-course introductory sequence and all other courses can be taken in any order. The result of this curriculum is that most courses contain relatively equal proportions of sophomore, junior and senior students. We expect that the less experienced students can learn from more experienced students on the same team. Therefore, we attempt to create teams that contain a mix of different experience levels. When relevant, we also distribute veterans of prior courses among the teams. Reflecting the size of our classes, teams are relatively small, consisting of 3 or 4 students each.

In Database Design, students were placed into teams prior to the course. The instructor chose teams according to the two criteria: mixing the traditional and weekend population, and providing each team with both experienced and less-experienced members. A third criterion for this selection of teams was to try to avoid putting very strong students on the same team with very weak students, because of prior problems with such teams.

In Project Management, students were allowed to choose their own teams. The only restriction on these choices was that each team contains at least one member from each population. Teams in Multimedia were self-selecting without restriction, since almost everyone in the class was a weekend student. Each team designated a leader, who served throughout the project.

In Interface Design, the instructor's original plan was to again allow the students to choose their own teams with the restriction that teams consist of both weekend and traditional students. However, the team project in Interface Design was to develop a web interface for an existing database. Therefore, additional expertise was necessary: each team needed someone familiar with databases, in particular the SQL query language, and each team needed someone who was comfortable programming in HTML and JavaScript. Because many of the students did not know each other before the course, the class requested that the instructor put together the groups. This was done according to the three criteria: every group must have a traditional member, every group must have a weekend member, every group must have someone with database experience, and every group must have someone with web development experience. Unfortunately, very few of the students who had completed Database Design took Interface Design, so not every team had a true database expert.

3.2 Assignments and Process

The assignments in each course required teams to work together on implementations, written documents, and a team presentation at the end of the course. Example assignments can be found at our web page, http://cs.hiram.edu/~walkerel/ccscne2002.html.

In Database Design, each team developed a database for an external customer. The project was divided into three stages, according to the process developed at the University of Arizona [10]. The course met for 7 weekends, with major deliverables on weekends 3, 5 and 7 and progress reports on weekends 2, 4, and 6. Weekend 7's deliverables included a presentation to the class and the external customers, as well as

a revised copy of all design documents, an evaluation of the project, and documentation for the customer. Team members took on assigned roles, which were rotated after each major deliverable. These roles included leader, recorder, checker, and technical expert [10]. Part of one class period was devoted to a discussion of team process, and teams were asked to report on their process during each interim report.

In Project Management, which met 3 weekends only, each team developed a requirements document and a project plan for a project that was loosely defined by the instructor. All deliverables (written project plan, requirements document, and oral presentation) were due at the end of the course. Teams selected their own team leaders, but no other role specifications were given. No specific instruction in team process was included in the course.

In Multimedia, which met for 7 weekends, each team developed a multimedia presentation using the Alice software. Teams designed their own presentations using storyboard techniques. Two teams used storyboard software they had located on the internet. No explicit instruction in the Alice software was given, although a 1-hour practice session was held during the 3rd weekend so that questions could be addressed by everyone in the class.

In Interface Design, which met for 3 weekends, each team was given a database with a minimal Web interface for which they developed and implemented an enhanced interface for a specified user. Although we had hoped to use the databases created in the Database Design course, technical difficulties required us to use new databases developed specifically for this course by a student assistant. Though no specific instruction in team management was given, teams were informed who had been assigned to the team as the database expert and who had been assigned as the web expert.

b. CHALLENGES OF TEAMWORK IN OUR ENVIRONMENT

Successful teams had to overcome not only the ordinary challenges of time management and learning to function in a team, but additional challenges due to our unique environment. These challenges included communication difficulties, which were exacerbated by the range of locations and lifestyles of our students, and differences among group members, which were more difficult to adjust to given the lack of regular class meetings. Throughout this section, issues and challenges are illustrated with excerpts from students' confidential group evaluations.

4.1 Barriers to Communication

One of the challenges that all teams faced was in communication. The nature of the weekend program is that students see each other in class when assignments are made and when deliverables are due, with one or two class meetings in between. Therefore it is crucial for teams to communicate well between classes. Both time and space present barriers to such communication, unfortunately. Traditional students live on campus and tend to work in the afternoon or late at night, while weekend students, living and working up to two hours from campus, do most of their work on their lunch hours, in the early evening, or on weekends. One weekend student, paired with a traditional student and another weekend student who lived far away wrote, “I do wish that we were not so far removed from one another because I feel that ... one-on-one interaction would really help me.”

These barriers of time and space are similar to those faced by multinational software development corporations, which address them using "video conferences, voicemail, electronic live-boards, the Internet and corporate intranets, groupware and virtual teamrooms" [1]. Our students used a subset of these tools, particularly e-mail and instant messaging software. Not all software was helpful. A group member wrote in her evaluation, "The logistics problems that we encountered with the online meeting not working out had me concerned..." The group reverted to sending documents to each other as email attachments. As a member reported, "we have communicated via email very regularly (almost daily)." The most successful groups used a combination of electronic and face-to-face meetings. The use of AIM (AOL Instant Messenger) software was initiated by one of the traditional students who used the software socially. A weekend student in his group wrote on her evaluation how glad she was to learn about a new type of software and how useful it was to the group.

Not all of the successful groups relied heavily on electronic contact, however. Two of the weekend students (one who was in two of the classes) made a point of driving to campus at least weekly for on-campus group meetings. The additional effort expended by these students was highly appreciated by other group members. Groups where the traditional students had cars met off-campus. Generally, groups that met face-to-face met two to three times between each pair of class meetings.

Groups that had difficulty with communication complained about "phone tag" and made little or no effort to find alternative means of contact. Instead, team members would either work independently (duplicating effort or worse), or do nothing until they got in touch with each other, even if this was the day before an assignment was due. In the worst case, a team broke in half, with the on-campus half and the off-campus half working on different versions of the project and attempting to fuse them on the last day, with a presentation that was obviously split. In her evaluation, a member of this team wrote, "By this point, the group has lost good communication... [team leader] thought [other member] and I were conspiring ...". Meanwhile, the team leader and the fourth member of the group were duplicating the other pair's effort.

Time for face-to-face meetings was also provided during classes. This was particularly important for a first meeting when groups were instructor-assigned. Even when no formal group time was included in a class, members of most groups would arrive early or stay late for a class in order to meet. This was particularly important on the day of presentation, when students needed to merge their work into a single coherent presentation. In the Interface class, by mutual agreement of the class and the instructor, the first half hour of the last class was allocated to final preparation of the presentations. This time was used not only for groups to coordinate, but also for setting up presentations for projection, e.g. copying them to the server. More than half of the students in the Interface Design class indicated that more group meeting time during class periods would have been helpful.

4.2 Diversity of Available Equipment

Another challenge that many groups faced was the diversity of available equipment. Unlike an on-campus project where students can easily use standardized equipment provided in labs, our off-campus students were limited to their own (or their employers') facilities. One group had significant difficulty passing their database around because it was too large for a floppy disk and one group member could only read and write ZIP drives, while another expected to pass information around on CD ROMs. To transfer information, this group had to meet at a lab at school and use the

local network. In one group in the database class, an off-campus student was effectively removed from database development because "by the time she was able to get Access and ftp ability, [we] were already well on our way with the forms and reports."

4.3 Disparity in Technical Background

Another challenge that teams faced was disparity in technical background among the team members. While the traditional students were usually junior or senior computer science majors with several programming courses and large projects completed, most of the weekend students had had only a single programming course and fewer courses where large projects were required. In all of our courses, teams were composed to take advantage of this diversity, hoping each member would learn from the others. Unfortunately, in a few cases, exactly the opposite happened: one student would decide that he (never a "she" in our classes) should do the whole project himself. A member of one such group wrote, "It would be nice if I could have assisted [team member] with a form or report but he wanted to do them all himself. I think he thought anyone else would just mess them up". As reported by a teammate, a member of another group discarded work done by his teammates without discussion: "[We] both made contributions to the user guide, which he had volunteered to put together. ...He, on his own, decided to withdraw information that I had [already put] into the guide."

Difficulties in communication and disparities in equipment and background can be overcome by a motivated group, or can strain a weaker group (especially one with non-team-player personalities) to the breaking point. Discussing some of the issues and providing technical help where possible (such as additional evening lab hours) have helped with overcoming some of these challenges. Others are best addressed by discussing them directly in class and with individual groups or students.

When possible, early intervention can bring a group back on track. For example, one group had difficulty with a member after the second week of the Database course. One group was "...greatly concerned about [member's] performance...[He] missed two of the four meetings ... [and when he] did attend meetings, he did not contribute... None of the group members are willing to trust [him] with any responsibility for the project." When confronted with these evaluations (and their expected effect on his grade), the student changed his attitude completely. In the final evaluation, this member's teammates noted his improvement: "[Member] made tremendous progress from his earlier reliability problems," wrote one teammate, while another wrote, "He has made a tremendous about face. [He] has shown up at every meeting...[and] proven he is a great Team Player!"

5 CURRICULAR EFFECTS BETWEEN COURSES

In addition to the general goal of promoting teamwork within courses, the curricular changes aim to develop a culture of teamwork throughout the curriculum. We expect that students will become effective team members, team leaders, and technical communicators through the sequence of courses. We hope that students will come to expect excellence in teamwork from their peers, and will live up to their peers' expectations. Finally, we hope that experienced students will begin to effectively pass the culture of teamwork on to the newcomers in their classes.

Some of these effects have already been seen. In Project Management, the selected leader of every team was a student with teamwork experience from Database

Design. Two student members of the least effective team in Database Design learned from their experience and formed the core of one of the most effective teams in Project Management. A member of the most effective team in Database Design brought his communication methods to become an extremely effective team leader in Interface Design. Students who had taken Database Design also served as effective database experts for teams in Interface Design. The most successful teams in Multimedia had members that had been in the earlier software engineering courses that emphasized teamwork. We expect that the trend of students carrying over both technical and process knowledge from course to course will continue this year, when students in Systems Analysis will augment the project plans from Project Management.

6 CONCLUSION

In our first year of emphasizing teamwork in our software engineering curriculum, we are already seeing a culture of teamwork develop. Experienced students are indeed improving the overall performance of their second and subsequent teams. Our graduates and weekend students tell us that the projects are realistic in their experiences. Many challenges remain, however. First among these challenges is the additional difficulty of communication given our class schedule and diversity of students. Second is the wide variety of available resources for our students, and third is the technical disparity between traditional and weekend students. Some of the changes that we can make to address these challenges are: (1) Explicitly address communication as part of the teamwork process, emphasizing available alternatives to the telephone or face-to-face meeting. (2) Allow additional in-class time for group meetings. (3) Make laboratory facilities available outside normal working hours. If it were possible to loan out laptops, external drives, or other relevant equipment, that would be even better. (4) Provide opportunities for early intervention wherever possible, to address personality or other difficulties before the group fails.

References

[1] R. Benson-Armer and T. Hsieh, Teamwork Across Time and Space, *The McKinsey Quarterly,* no. 4, 1997, 19 – 27.

[2] J. Brown and G. Dobbie, Supporting and Evaluating Team Dynamics in Group Projects, in *Proceedings of SIGCSE* 2000, (February 2000) 40 – 44.

[3] T.B. Hilburn, Software Engineering Education: A Modest Proposal, *IEEE Software* (1997).

[4] J. Kiper, M.J. Lutz, and H.A. Etlinger, Undergraduate Software Engineering Laboratories: A Progress Report from Two Universities, in *Proceedings of SIGCSE* 1992, (1992) 57 – 62.

[5] P.K. McKinlye, B.H.C. Cheng, and J.J. Weng, Moving Industry-Guided Multimedia Technology into the Classroom, in *Proceedings of SIGCSE* 1999, (1999), 160 – 164.

[6] W. Mitchell, What I have Learned in Twenty Years of Teaching the Software Project Course, in *Proceedings of the CSCC Southeast Conference,* (1998).

[7] M. Osborne, Software Engineering, C++, and Windows, in *Proceedings of SIGCSE* 1996, (1996), 243 – 247.

[8] J.A. Preston, The Real World Lab: A Software Engineering Practicum, 1998. Online Internet. http://www.cc.gatech.edu/classes/RWL/Web/rwl_description.html,

[9] I. Somerville, *Software Engineering, 5th ed.*, Addison Wesley, 1998.

[10] S.D. Urban and S.W. Dietrich, Integrating the Practical Use of a Database Product Into a Theoretical Curriculum, in *Proceedings of SIGCSE* 1997, (1997), 121-125.

[11] E.E. Villarreal and D. Butler, Giving Computer Science Students a Real-World Experience, in *Proceedings of SIGCSE* 1998, (1998), 40 – 44.