Report: A Capstone Project involving a Hundred Students, for an Industrial Partner

Authors:

Daniel Stearns, California Polytechnic State University, dstearns@csc.calpoly.edu
John Dalbey, California Polytechnic State University, jdalbey@calpoly.edu
Clark Turner, California Polytechnic State University, csturner@csc.calpoly.edu
Tim Kearns, California Polytechnic State University, tkearns@csc.calpoly.edu

Abstract — *The recently approved Cal Poly software engineering major concludes with a significant capstone project. During a yearlong course sequence, students specify, design, construct and deploy a large software project for an industrial partner. This report summarizes the capstone sequence project in the current academic year; sixteen student teams constructed an automated testing system for an industrial partner - Brocade Communications. The teams confronted problems typical of a large, real-world software project: revolving requirements, user interaction, resource conflicts, team management and deployment issues.*

Index Terms — *University-industrial collaboration, capstone course, project-based education, software engineering degree*

BACKGROUND

California Polytechnic State University (Cal Poly) was founded a century ago with a special mission - theoretical knowledge is best learned through its application to practical problems. The Cal Poly motto "*Learn by doing*" is widely quoted and permeates all degree programs. Cal Poly is a polytechnic institution with several nationally ranked engineering programs that graduate a thousand engineers yearly. Each graduate completes an extensive general education program combined with study of a specific engineering discipline. Graduates enter their careers with a practical understanding of their profession due to a series of lab and real-life experiences that apply theory to real engineering problems.

The Computer Science Department hosts three undergraduate majors – Computer Science, Computer Engineering and the recently approved Bachelor of Science in Software Engineering. All three majors prepare students for professional careers in software and hardware development. A large majority of the graduates enter the workplace immediately after graduation.

The distinguishing features of the software engineering major follow from the observation that software engineering deals with deployment-oriented software development. These features are unique to the software engineering major:

- Development of an actual software product
- Scheduled and timely delivery of that product
- Delivery of the product under resource constraints
- Customer satisfaction with the product

The software engineering major addresses those features with a focused set of five software engineering courses. [Figure 1 and Table I]. An initial two course sequence is offered in the second year to introduce software engineering principles while students construct a sponsored software product. In the fourth year, in a three course capstone sequence, students study advanced software engineering and lead a team to develop a software product.

Capstone Project Criteria and Choice of Industrial Partner

The software engineering faculty selected the project based on the following criteria:

- There must be a working relationship between the industrial partner and the Computer Science Department
- The industrial partner must commit to a year-long relationship

- The project size must be in the range of 500-1000 function points [2]
- The project must be constructed as a Java application

Brocade as Industrial Partner

Cal Poly's Computer Science Department has strong ties to the California computer industry, primarily in Silicon Valley. Several companies were interested in sponsoring the capstone project. Brocade Communications, Inc. of San Jose, Californa was selected because their project best met the capstone project criteria.

Brocade Communications, and other industrial partners, provide projects to Cal Poly for a number of reasons:

- Corporate visibility - a large number of Cal Poly students become familar with the company and its products.
- Recruiting - California companies compete vigorously to hire the best university graduates; students in the capstone sequence are excellent employee candidates.
- Useful but low priority project - the project is of interest to Brocade because they lack the resources to work on automated test tools. Cal Poly students do the primary research, create basic algorithms and construct a working program.

Except for an intellectural property agreeement, there was no contractual nor monetary arrangement betweeen Cal Poly and Brocade. Students performed the work on a best-effort basis; their work became Brocade's intellectual property.

Paper Overview

This paper is organized by the chronological sequence of the capstone courses. The faculty negotiated the intellectual property (IP) agreement in the term prior to the outset of the capstone sequence; that negotiation is discussed. Then results from the three capstone courses (*Requirements, Construction, Deployment*) are reported. The student's view of the courses are summarized with an emphasis on the leader viewpoint. Finally, the paper discusses conclusions and plans for next year.

INTELLECTUAL PROPERTY ISSUES

The cornerstone of the capstone sequence is the industrial project. In order to entice a potential industrial partner, the project must have some real value. The project wasn't on a critical path or even particularly important for Brocade; that would be infeasible for an university course. Rather, the results would be viewed as a prototype that might give Brocade useful feedback about a potential future product.

In any case, Brocade insisted on full ownership of all intellectual property created at Cal Poly. Brocade would own the results and could potentially use them in a future product or to demonstrate potential for future work.

Intellectual Property issues can become contentious; normal practice is to negotiate an agreement that explicitly states the rights of each party. In this case, there were four parties: Cal Poly, the students, the software capstone faculty and Brocade.

Cal Poly doesn't have an office to negotiate such agreements, so the faculty negotiated the agreement with Brocade and enlisted the Cal Poly Dean of Research for support and approval.

The negotiation process required significant faculty time over several months. There were a number of telephone conferences with Brocade's attorney; it was often necessary to remind him students would be constructing the software in a college course under no contractual agreement. The attorney wanted assurance that the intellectual property could be used freely in any Brocade projects. He wanted Cal Poly, the capstone course faculty and the students to yield any future legal claims to the work.

Cal Poly and the faculty signed the agreement to yield ownership to Brocade. The students were asked to sign a similar agreement as a condition of full participation in the course. Some of the students balked while asking questions of a legal nature. The explanation was straightforward: Brocade would own all intellectual property created in the course. A small number of students refused to sign; there was no process for dealing with such students and they remained in the capstone sequence.

PROJECT DESCRIPTION

The capstone project was an automatic test case generation and execution tool (CAPATT - Cal Poly Automated Test Tool). Brocade's major product line are switches, hardware and software comprising a Storage Area Network [1]. One software product, currently under development, is an industry-standard data management framework. Verification of this product is not complex but requires manual construction of a huge number of system test cases. The capstone course project was a utility that automatically generates those test cases, executes them and verifies the results.

REQUIREMENTS ELICITATION COURSE

The *Requirements* course deliverable was a Software Requirements Specification (SRS), software architecture and prototype that can be used by the *Intro SE I* students to learn the problem domain. Students were organized into five teams; student grades were based on the team process, individual contributions to the team process and quality of final deliverables. There was only a single course and single instructor so the grading scheme was straightforward - unlike the *Construction* course grading scheme described below.

Brocade Interaction and Responsibilities

During the *Requirements* course, Brocade was reponsible for providing domain information and tools for team use. Brocade personnel travelled to Cal Poly at the outset of the course in order to give the student teams an introduction to the problem domain, to the problem and to deliver basic documentation. They were available by email and telephone through the course instructor. The instructor and selected team representatives made four visits to Brocade during the course to obtain feedback and information. This face-to-face contact between the students and Brocade professionals resulted in many technical insights and increased student morale.

Software Requirements Specifications (SRS)

All five SRS documents contained considerable design bias. This is not unexpected; most students and many professionals have a difficult time separating the problem domain from the solution domain. Many students did not recognize the importance of problem domain thinking; as a result, problems arose in the *Construction* course described below. The design bias could also be an effect of the compressed ten week course which is a short time to understand a complex new domain.

Results from a previous capstone sequence were reported at ICEE 2001 [3]. At that time, there were three industrial partners, each with a distinct problem domain. Serious problem domain issues arose at that time; in an attempt to remedy the problem, one industrial partner (Brocade) was used this year. But, again, the problem domain was too complex and required more time and effort than had been anticipated.

Some new problems arose with five independent teams specifying the same product for a single client. Consequently

- If the Brocade representative was busy or missed a deadline, all the teams experienced delays and frustration.
- Requirements elicitation meetings and interviews were too large and unwieldly.
- The sharing of common material and information became a problem due to competitive pressures among the teams. Students perceived that withholding information might give them a competitive advantage over other teams.

SOFTWARE CONSTRUCTION COURSE

Most of the *Construction* course students served as leader of a team charged to construct CAPATT. A few students assumed specialty leaderships roles in areas such as Quality Assurance or Database Administration. The project portion of the software construction course had a simple objective - build a product that can be installed at the customer site.

Course Organization

The *Construction* course is taught in parallel with the second year *Intro SE I* course [Figure I]. The parallel courses share a common laboratory space and time; each student team contains students from both courses. Each fourth year students assumed leadership roles directing a team of second year students. This provided the leaders management experience with an apprentice learning style for the second year students.

Teams constructed the CAPATT project with several major deliverables such as a feasibility prototype, system test plan, software design and staged implementations. Much of the management and direction of the teams was the responsibility of the team leader. The instructors imposed deliverable and schedule constraints but did not supervise each individual team.

Brocade Responsibilities - Construction Course

In the *Construction* course, Brocade's primary reponsibility was to perform design reviews. The Brocade personnel visited Cal Poly, reviewed all sixteen team designs and provided feedback to the students.

Grading Scheme

Many American university students place a high emphasis on their course grades. The creation of an appropriate grading scheme in a team-based project course is a challenge; it is more complex when team members are graded by two different instructors. Several schemes had been discussed during preparations for the capstone sequence; the faculty and students spent much time discussing an appropriate grading scheme. The grading scheme used during the construction course, is described:

- The project was a substantive portion of the course grade
  Work on the CAPATT project was the largest component of the course grade for the *Intro SE I* students. This did get their attention; there wasn't a single student who ignored the project in order to master the lecture material. Unfortunately, many students believed the project grade comprised their entire course grade; this caused some confusion.
- Team leaders were given significant grading authority
  At the end of the course, team leaders assigned each individual team member a project grade. The instructors used the assigned grade at their discretion; the leader grades were at least a strong recommendation. In any case, the *Intro SE I* students perceived their leader as their evaluator; this was the grading system goal.
- Deliverables and milestones were assigned by faculty
  It was deemed infeasible for the team leaders to determine the team deliverables and milestones; the instructors determined the team work products and due dates.
- Team leader evaluations by each individual team members
  At the end of the course, each team leader was evaluated (only for her leadership). This evaluation had a direct affect on the leader's grade and was not subject to instructor discretion.
- Subjective project grade for the team leaders
  The *Construction* course instructor assigned a project grade to each leader based on his or her work. This grade was assigned subjectively using a self-evaluation, status reports and team acceptance test results

Course Problems

Several significant problems emerged during the *Construction* course.

- Problem domain mastery
  The *Intro SE I* students took much longer to master the problem domain than was anticipated. Substantial lecture time was devoted to discussing the problem domain; an examination on the problem domain was given during the third week. Even so, many students didn't fully understand the domain until the end of the course. Some students never did understand the problem CAPATT was intended to solve.
- Inadequate Software Requirements Specifications (SRS)
  The SRS documents were too large, incomplete, failed to distinguish functional from non-functional requirements and confused design issues with analysis issues.
- Lack of useful functional prototype
  The *Requirements* course students created prototypes better characterized as storyboards; they failed to demonstrate CAPATT's basic functionality. Useful prototypes weren't completed until the midpoint of the *Construction* course.
- Inadequate software process or software process knowledge
  The team leaders were expected to provide software process guidance for their teams. For a variety of reasons (see feedback sections below), this only happened in a few of the sixteen teams.

SOFTWARE DEPLOYMENT COURSE

As the final course in the capstone sequence began, four of sixteen CAPATT implementations were installed at Brocade. The sixteen *Construction* course teams were consolidated into six larger teams to work on the four delivered products plus two other products worthy of continued work. The work consisted of the following type of tasks:

- Repair product defects
- Build install/uninstall systems
- Add better support for configuration parameters
- Add minor functionality
- Write scripts to perform high-quality system and coverage tests

Brocade Responsibilities - Deployment Course

A major aspect of Brocade's commitment to the capstone sequence is their evaluation and use of the installed products. Their test group is expected to devote substantive time to the CAPATT project.

Problems

It is infeasible for students to construct an important industrial project as part of their coursework. At the same time, the deployment process requires that the partner expend resources to use and evaluate the products that have been installed at their site. The dilemma: it is difficult for the customer to devote resources to a low priority product but the students can't feasibly work on a higher priority product. This appears to be a dilemma without a solution.

TEAM LEADER FEEDBACK

A formal focus group was conducted with the team leaders at the beginning of the *Deployment* course; none of the course instructors were present during the session. The focus group goal: to understand and document the leader's perceptions of

- The technical content of the courses: process models, tools, design concepts, test plans, quality assurance.
- The team and management content of the course: leadership and organizational skills
- The major problems they encountered as team leaders
- Aspects of the course that were conducive to learning; aspects that inhibited learning

In general, the leaders expected to have more control over team management, the software processes used and the scheduling of deliverables. At the same time, they wanted a more supportive environment. Specifically, they mentioned:

- The *Construction* course lecture should focus, early in the term, on leadership skills and provide a venue to discuss management problems. Leaders felt a wider range of case studies would have helped them with the problems they faced.
- A desire for more consistency in management philosophy between instructors, especially with expectations for what was important for deliverables and grading. This tension was particularly acute in the importance given to the final product.

The team leaders wanted more guidance and structure, but also permission to deviate from the guidance provided they could still achieve the course objectives. In particular, leaders wanted more instruction on their roles and how to best meet the demands of their roles. For example, they felt they did not have a good working knowledge of software processes at the beginning of the *Construction* course. This made it difficult to choose an appropriate process model and team structure.

Another problem was the perceived instability of requirements. The leaders understood that changes are a normal part of industry projects. But they believed that stable requirements would have provided a better learning environment. They also felt the problem domain required too much time to learn.

The leaders believed that the *Intro SE I* students learned material equivalent to a more traditional course. But, the students could have learned more, with less frustration, had the course lectures been better synchronized with the project. The leaders believe the course lecture should have its own clear objectives, linked to the goal of completing the project.

The leaders did not see the connection between imposed deliverables and achieving their team goals, as they understood them. Consequently, some deliverables were viewed as extraneous given the real work of producing a working product.

Leader feedback - conclusions

These reactions reflect the tension in meeting conflicting course goals. The capstone sequence courses must provide a realistic simulation to motivate the students while they confront the many technical and personnel issues that arise. At the same time, students must acquire knowledge of the processes and practices that comprise a software engineer's skill set.

The course instructors have had extensive management experience. It is interesting to note that the leader's perceptions are quite similar to those of first time managers in any large software development organization. To wit, most first time managers desire more guidance and resources but also desire more autonomy.

The leader's experience indicates that one course goal was admirably achieved; there was a fairly accurate simulation of the real world experience of software engineers and managers. The traditional learning objective, to provide students with software engineering knowledge, can only be assessed by evaluating the success of the students as software engineers.

STUDENT FEEDBACK

Intro SE I students wrote a weekly status report; at the end of the course, they wrote a self-evaluation which included what they learned from the course. In addition, some students were debriefed at the end of the course. There was no shortage of student opinions regarding their experience.

- Team interactions
  CAPATT was the first team project for most *Intro SE I* students; for many it was the first time in their lives they cooperated for an extended time with peers. Students had a difficult time with team dynamics citing issues such as poor communication, many wasted meetings, inadequate planning and hero reliance. Many learned the most from these issues with which they struggled.
- Time
  Students logged their hours in a time log. Almost all of the students worked an excessive number of hours on the course; most believed there was too much required work. Many students were unable to understand the nature of an open-ended project such as CAPATT.
- Assignment conflict
  Students were assigned tasks both by their team leader and by their course instructors. They perceived conflicting tasks, due dates and expectations and had a difficult time making task priority decisions.
- Technical issues
  Students appreciated the technical material they learned in the courses. Examples noted by students included advanced Java programming skills, software design, UML and work on a large project with a complex build.
- Software processes
  Almost all software engineering students resist imposed software processes. At the end of *Intro SE I*, students recognized the value of some of the processes they resisted during the course such as clear and complete requirements, formal technical reviews, test plans and configuration management practices.

CONCLUSIONS AND PLANNED CHANGES

A year long project involving a hundred students, three faculty members and an industrial partner generated substantive learning. Every student wrote a weekly status report; teams met regularly with faculty members to discuss issues and there was much interaction between students and faculty. One issue dominated the capstone courses - process versus product.

Tradeoff: Balancing Process and Product

The major challenge of teaching a project-based course is that students will focus on project construction; they tend to neglect the educational goal - to study software engineering practice and processes. Students are familar with "programming in the small" from their previous courses. They, quite naturally, tend to attack a larger problem with their extant skills.

The Brocade project was far larger than most students had ever attempted, yet not so large as to be overwhelming. The solutions were in the 5-10 KLOC range. It is likely that a few talented student programmers could have constructed a reasonable solution, with little software process, during the *Construction* course. Many of the student leaders realized this and wanted to construct the project with minimum software process.

Instructors wanted students to study software processes even though some process steps would not be required to achieve project success. Students viewed process work as an intrusion on the "real" coding work. Many academic subjects present similar challenges; in this case, the student's disdain for some deliverables was particularly acute. This was likely due to the corporate nature of the capstone project; students felt compelled to create a working product.

Based on this experience and the student feedback, the software engineering faculty have made the following changes for the coming year. This paper is submitted prior to the completion of the *Deployment* course; final conclusions and decisions will be reported at ICEE 2003.

No corporate project

Even though the Brocade Communications personnel embraced the project and provided adequate resources, there were some difficult problems with the industrial partnership (1) Brocade is located two hundred miles from Cal Poly; there was little opportunity for face-to-face communication. (2) The users, as expected and desired, continually "discovered" new requirements and altered existing requirements. (3) Even though the project domain was thought accessible to the Cal Poly students, they had a difficult time understanding the domain, its tools and its requirements. (4) The intellectual property agreement was a time consuming effort.

Consequently, the faculty have decided to teach the capstone sequence next year without an industrial partner. Instead, the students will construct a software project for a Cal Poly organization with a software engineering faculty member serving as the project user. This decision will also eliminate any intellectual property concerns.

Course realignment

The students in the *Intro SE I* course were given a Software Requirements Specification and expected to learn the problem domain quickly; this did not happen! Most of the students required several weeks to understand the project requirements and

become familiar with the domain. Numerous discussions and status report readings led to an obvious conclusion - it is infeasible for second year computer science students to learn a complex domain in a short time. As a result, the course alignments have been changed; *Intro SE I* will be aligned with the *Requirements* course and *Intro SE II* with the *Construction* course. With this realignment, the *Intro SE I* students will participate in requirements elicitation in order to understand the problem requirements before they construct the software in the *Intro SE II* course.

REFERENCES

[1] Beauchamp, C. and Judd, J. *Building SANs with Brocad*e, Syngress Press, 2001

[2] Garmus, D. and Herron, D., *Function Point Analysis*, Addision-Wesley, 2001

[3] Stearns, D. , Meldal, S., Turner, C, "Ten Pounds in a Five-Pound Sack", ICEE 2001

FIGURES AND TABLES

TABLE I

SOFTWARE ENGINEERING COURSES

Course	Abbreviation used in text (in italics)
Introduction to Software Engineering I	*Intro SE I*
Introduction to Software Engineering II	*Intro SE II*
Software Requirements Elicitation	*Requirements*
Software Construction	*Construction*
Software Deployment	*Deployment*

FIGURE 1 - CAPSTONE COURSE STRUCTURE (2002/2003)