Project-centered Multi-Course

Patricia Lasserre[i],
Associate Professor, Computer Science Department, Okanagan University College
And
Yves Lucet[ii],
Assistant Professor, Computer Science Department, Okanagan University College

Abstract: The findings and recommendations based on our experiment on a project-centered multi-course are detailed. Our experiment used a software engineering course as the core with several courses gravitating around. The idea was to use the gravitating courses to speed up project completion while giving students first-hand experience on a medium-size project and on people issues.

We present our objectives, detail the experiment, analyze our findings and make recommendations for improvements.

Introduction

The objectives for expanding our software engineering two-course sequence into related courses taking place the same term were as follow:

- To reduce the amount of background materials the student must learn before they can work on the core concepts. Using the same project as background saves the work to recreate artificial examples to explain specific part of the course and enables the student and the instructor to focus on the new core information only without being distracted by the specific details of an example.
- To provide the students with a management experience involving a much larger team (15 students) than in our regular project course (between 3 to 8 students).
- To tackle most of the facets of a large project in an enterprise-like setting.
- To provide an opportunity to find a role that fits students with little programming or database background.
- To go beyond small academic assignments and get a closer experience on a real-world project. In particular, to become comfortable learning on the job new technology.

Experiment

Background

Okanagan University College offers a diploma program in Computer Science the Computer Information Systems diploma, and two bachelors in Computer Science: the bachelor of Computer Information Systems and the Bachelor of Science with a major in Computer Science. The software engineering courses used to carry on our multi-course experiment are *COSC 310 - Software Engineering* and *COSC 319 - Software Engineering Project*. The courses go together: *310 - Software Engineering* is a pre-requisite for *319 - Software Engineering Project* and students usually take *310 - Software Engineering* in the fall followed by *319 - Software Engineering Project* in the winter except on rare cases e.g. when a student fails *310 - Software Engineering* or when a student takes a co-op position in the winter.

Students taking *310 - Software Engineering* have a diverse background. Some already have the CIS and have taken our project course *COSC 224 - Project in Computer Science*. Others are working towards the completion of a BCIS or a BSc (not necessarily with a major in Computer Science). Hence their skills vary widely between very good programming experience and having already taken *COSC 224 - Projects in Computer Science* for a CIS graduate to very little programming experience and no project completed for a BSc student working towards a major in Mathematics. The wide range of skills has historically been a challenge for the instructor and was one motivation to experiment a new format for the course. The other motivation was to introduce Agile methodologies and to update the content of *310 - Software Engineering* by starting the project much earlier than in preceding years.

In the past *310 - Software Engineering* focused on the theory of software engineering with no implementation of the project. At best students started to gather requirements then implemented different projects in small groups in *319 - Software Engineering Project*. The *319 - Software Engineering Project* course was then similar to 224 - *Projects in Computer Science* but focusing on bigger projects. The challenge has been to introduce students with little knowledge of programming to large projects. In the fall 2002, *310 - Software Engineering* students started a common project for the whole class. They also started implementation in *310 - Software Engineering* instead of *319 - Software Engineering Project*.

In the fall of 2003, the *310 - Software Engineering* course was linked to *COSC 305 - Project Management, COSC 341 - Human Computer Interaction*, and to a lesser extend to *COSC 404 - Database System II*. No requirement (pre-requisite, co-requisite) was imposed so some students took all 3 courses, while others only took one course. We will discuss below the challenge of keeping efficient communications.

The project selected was to build a client-server system to facilitate the scheduling task of the Computer Science department chair. It involves creating a user interface, a database, and some server-programming layer. In the fall 2002, the *310 - Software Engineering* class started working on the project while the project was used as a course scheduling assignment for *MATH 441 - Modelling of Discrete Optimization Problems*.

Several faculty members were involved. The department chair was the customer for the project. He reviewed the user interface and the requirements to make sure the project would meet his expectations. The instructor for *305 - Project Management* and *341 - Human-Computer Interaction* interacted with the chair to get feedback on the student interfaces generated in *341 - Human-Computer Interaction*. She also coached the *305 - Project Management* students to gather all the requirements necessary to create a schedule for *310 - Software Engineering*. The instructor of *310 - Software Engineering* and *404 - Database Systems II* helped the *310 - Software Engineering* students manage the project. He also created a lab in *404 - Database Systems II* to speed up the refactoring of the database code generated the year before.

The multi-course setup

The students *in 310 - Software Engineering* were grouped into teams. Five teams were created: Programming I, Programming II, Database, Testing, and Administration (management and server setup). Each team had a team leader reporting to a project leader. Additionally, one student was also a backup project leader for the times when the project leader could not make it to the lectures.

The *341 - Human-Computer Interaction* students were split into several groups. Each group had to develop a prototype graphical user interface (GUI) for the project. Two families of prototypes were considered: Web based client and non web-based clients. Every iteration, the class had to gather requirements, create a new prototype, and then have it reviewed by the client.

The *305 - Project Management* students were to gather the requirements to be implemented in *310 -*

Software Engineering, generate a schedule, and assign weekly tasks for the *310 - Software Engineering* students. The project leader student in *310 - Software Engineering*, who was also taking *305 – Project Management*, was responsible to summarize the *310 – Software Engineering* class progress and report it to *305 – Project Management*.

The *404 - Database Systems II* class only spent two weeks working on project-related assignments. They created PostgreSQL SQL and PL/pgSQL procedures to isolate access to the database and to implement constraint checking. These functions sped up the refactoring of the database code considerably. They also provided needed examples to *310 - Software Engineering* students who had little previous database background.

One of the key components of the multi-course is the communication between the different courses. Since not all students take all the courses involved, some means of keeping everyone up to date was needed. Three communication processes were setup:

- At the faculty level, both instructor involved exchanged information on the project on a daily basis and the *341 - Human-Computer Interaction* instructor had weekly meeting with the customer/chair. The chair also attended two lectures of *310 - Software Engineering* and *341 - Human-Computer Interaction* to provide additional feedback on some requirements.
- At the course level, the project leader was required to write weekly reports for *310 - Software Engineering*, to summarize these, and bring them to *305 – Project Management*.
- At the student level, students unfamiliar with the project were encouraged to talk and extract the information from the students already familiar with the project.

The results

Student feedback

At the end of the term, the *305 - Project Management* students filled in a survey questionnaire to provide insight into their perception of the project centered experiment. They were asked to identify malfunctions and to provide suggestions to improve the experiment. The following malfunctions were noted:

- Some previous *310 - Software Engineering* students felt badly about their project being continued. In essence, their work was now judged by the new *310 - Software Engineering* students who had to work from it. Code ownership was not so much an issue as the feeling new students criticizing their work. The feeling was amplified by some requirements changes that occurred after the first course was delivered but before the next one started.
- Some assignment deadlines were kept although the software in the lab was altered decreasing the time usually given to complete assignments.
- Change of requirements between the fall 2002 and the fall 2003 of the *310 - Software Engineering* class confused students. Moreover, requirements changes during the course of the term weakened greatly the student management predictions.
- Students were not accustomed to manage a project and run into common management challenges that made their schedule widely off target. Managing their stress became more challenging than in regular courses. Some students indicated that students should not control other students because they had no authority to do so.
- Students could get answers from three faculty members. Sometimes they got different answers from the customer (chair), the project management instructor, and the software engineering instructor. They realized the answers were different and complained about it. They also realized we did not

give different answers on purpose. Requirements were too vague to provide unambiguous interpretation.

- Students complained that missing a deadline had no real consequence. They were confused at first by an iterative delivery model. In the end, the software engineering instructor had to assign specific tasks to speed up the implementation.
- The reduction of small examples to illustrate concepts was felt, with some students feeling thrown on a large project too soon while they would have preferred getting more smaller assignments.
- Chaos was felt especially at the beginning of the term. It was mostly due to the students' lack of management experience e.g. it took several weeks before the project leader realized that teams can work in parallel and not everything follows the waterfall model.
- Students quoted Brook's law[1] several times during the term to justify their difficulties. It took them several weeks to realize that some tasks can be done in parallel like coding independent modules.

Faculty feedback

Faculty members made the following observations:

- A modification of the pre-requisite/co-requisite of the course involved would allow a much more homogeneous student population. In particular, it may be possible to have most of the students taking the same courses thereafter creating a team structure earlier as well as a common set of skills.
- Many issues the students struggled with are similar to real-world software development issues. For example, communication issues, absenteeism, unreasonable deadlines, lack of technical support, unclear requirements, multiple customers with conflicting views and objectives... Consequently, the instructors saw the frustration of some students as part of the teaching process. They did not add more challenges. They just left the students with fewer directions than in ordinary courses.
- The creation of the teams followed closely the four phases (forming, storming, norming, performing) as described in ***"Successful Project Management", 2nd edition,*** *by Gido and Clements*. It was revealing how students embraced the concepts of the four phases when presented at the end of the semester.
- Most of the issues were people issues that no software process can solve. Although the more mature students realized it, most thought that with clearer requirements, better technical support, more knowledge and reasonable deadlines, the project would be a clear success. Unfortunately, this ideal situation almost never materializes in practice. It was very interesting to observe the students learning the people aspect of projects instead of the usual technical aspects.
- The instructors noticed and fought over a passive attitude behavior in certain students. Apparently students get accustomed to simply follow instructions in most courses. They then failed to take initiatives in our very open multi-course format. A wait and see attitude put the project off schedule at the very beginning of the term.
- The instructors also noted the time management behavior of the students. Even motivated students refused to spend large amount of time every week to balance their workload between all the courses

---

1"Adding manpower to a late software project makes it later" -- a result of the fact that the expected advantage from splitting development work among N programmers is O(N) (that is, proportional to N), but the complexity and communications cost associated with coordinating and then merging their work is O(N^2) (that is, proportional to the square of N). The quote is from Fred Brooks, a manager of IBM's OS/360 project and author of "The Mythical Man-Month". Quoted from Eric S. Raymond (compiler), New Hacker's Dictionary (The MIT Press, Cambridge, M.A., 1996) as listed at http://jamesthornton.com/theory/theory?theory_id=27

they were taking. This time juggling appears much more clearly than in previous years. Clearly students expected less homework with a gap between the instructor and the students on the weekly workload. For five weekly contact hours in *310 - Software Engineering*, students worked 3 hours on average instead of the expected 5 hours; a 40% difference.

- Working in parallel and modularity was a surprisingly difficult concept to get through. While Brook's law states "Adding manpower to a late software project makes it later", he noticed that it is an oversimplification adding: "the maximum number of men depends upon the number of independent subtasks"[2]. Students struggled to assign independent tasks most likely by lack of project management experience.
- The communication structure for the *310 – Software Engineering* and *305 – Project Management* was long to establish for the students. It reduced students' motivation and absenteeism in *305 - Project Management* forced the instructor to regularly improvise new teams, or reorganize existing ones when students came back. It also implied that the participating students had to create a schedule with insufficient data. The instructors regarded the issues positively: the instructor and the students had to work on solutions instead of focusing on problems. It taught the students skills and attitudes they will need when they enter the work force.
- While *341 - Human-Computer Interaction* was not dependent on *310 – Software Engineering* course, it had its own challenges. First, the ambiguity of the requirements created frustration for the students. In particular, students that had initially worked on the project the year before were reluctant in looking at a different interface. However, we believe the project benefited from the experience: requirements were much more defined, even changed after realizing how the customer would use the software. For example, it was realized that the customer should be able to save several schedules for the same year. Initially, they assumed that one schedule should be enough.
- Coordinating the different courses between the two instructors involved was a challenge. Daily (unstructured) information exchanges took place and even then some information was lost. Scheduling several short weekly meetings between instructors involved, or some other form of keeping the instructor team up to date appears necessary.
- Students were evaluated in their respective courses following a strict separation of tasks between the courses. With very few exceptions students were clearly aware of the association between specific tasks and its course. Communication between instructors was the key to prevent students from getting credit twice for the same work.

Conclusion

The instructors felt that globally the students benefited from the experience. A number of points to improve were identified:

- Harmonizing the student population by enforcing pre-requisites and co-requisites in the courses involved e.g. the *342 - Human Computer Interaction* course could become a pre-requisite to force the students to gather precise requirements through the completion of an interface. Similarly, the *305 - Project Management* course could become a pre-requisite to force the students to create a schedule as complete as possible before the project implementation starts. Alternatively, keeping the *310 - Project Management* course as a co-requisite allows the students to manage a team of developers as the project is being implemented.
- Enhancing the communications among the students. Although students responsible for the

2 The mythical man-month, Frederic P. Brooks, Jr., Anniversary edition, Addison Wesley, 1995.

communication were clearly identified they could have done a much better job at keeping everyone informed. While it appears as a people issue more than a process deficiency, stronger incentives may help.

In the following term (winter 2004), the *319 - Software Engineering Project* course implemented some suggestions. The lack of consequences for missing deadlines prompted the instructor to implement a nonstandard course evaluation scheme. The class was given a common mark based on the number of features implemented, the quality of the code produced, and more generally what the project produced. Then individual modifiers are added or subtracted to take into account the variation of productivity, and effort between students.

After completion of the course, the instructor plans to revise the evaluation scheme for the winter 2005 as 60% common class mark and 40% personal modifier instead of 100% class mark +/- modifier. The student modifier will be partly based on a student report on his achievements.

[i] Patricia Lasserre, Associate Professor, Computer Science Department, Okanagan University College, 3333 University Way, Kelowna, BC V1V 1V7, Tel. (250) 762-5445 ext 7500 plasserre@ouc.bc.ca

[ii] Yves Lucet, Assistant Professor, Computer Science Department, Okanagan University College, 3333 University Way, Kelowna, BC V1V 1V7, Tel. (250) 762-5445 ext 7891 ylucet@ouc.bc.ca