9-2007

A Classroom Outsourcing Experience for Software Engineering Learning

William L. Honig
Loyola University Chicago, whonig@luc.edu

Tejasvini Prasad
University of Wisconsin

A Classroom Outsourcing Experience for Software Engineering Learning

William L. Honig
Loyola University Chicago
Department of Computer Science
Chicago, Illinois 60611 USA
+1.312.915.7988
whonig@luc.edu

Tejasvini Prasad
University of Wisconsin
Dept. of Mathematics, Statistics, and Computer Science
Menomonie, Wisconsin 54571 USA
+1.510.299.2106
teju.prasad@gmail.com

ABSTRACT

Outsourcing of software development is a key part of globalization, oft misunderstood by computer science students, and possibly a cause of declining enrollments in the field. The authors developed and implemented an outsourcing experience for students in an advanced software engineering course. Student teams at two universities developed game playing programs and outsourced key parts of their systems to the other university. Results show students improved their understanding of outsourcing, developed better appreciation for the importance of software engineering techniques, and created ad hoc communication protocols between teams. The paper concludes with recommendations for expanding the approach used to other universities to create a more inclusive computer science and software engineering teaching environment.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management – *life cycle, programming teams, software configuration management;* K.3.2 [**Computers and Education**]: Computer and Information Science Education – *curriculum, information systems education;* K.6.1 [**Management of Computing and Information Systems**]: People and Project Management – *life cycle, management techniques staffing.*

General Terms

Management, Measurement, Experimentation.

Keywords

Outsourcing, software engineering, global software engineering, team communication, software development life cycle, computer science enrollment.

1. INTRODUCTION

This paper reports the results of an experimental undergraduate course to introduce outsourcing of software development into the software engineering curriculum. Section 2 motivates such experiments from the impact outsourcing is having on the field. The course structure, including having teams at two universities both do outsourcing for each other, is described in section 3. The course changed the student's perceptions about the benefits and evils of outsourcing as well as successfully teaching key software engineering concepts (section 4). Section 5 recommends further work to include outsourcing experiences for students as a way to enhance their understanding and readiness for careers in the global software development field.

2. RELEVANCE OF OUTSOURCING

Outsourcing is widespread in many industries and fields today, including systems and software development. Many global organizations perform software outsourcing internally between different divisions and large firms exist to do outsourcing of software development for others. Hence, it seems essential for computer science education to look for ways to introduce learning about outsourcing into the curriculum.

The authors believe that the concern about outsourcing eliminating software jobs (or reducing the compensation potential) has contributed to a decline in computer science enrollments in universities (there are likely other causes as well). One of the students enrolled in the course described in this paper summarized a common opinion about outsourcing in a semester start survey: "I don't like outsourcing because of the adverse effect it has on my future job security. Outsourcing will work best when both programmers have equal wages (U.S. current wages)". Similar concerns seem very common among today's students.

The author's work in creating the course concept was not motivated by a fixed belief that outsourcing is "good" or "bad", or has a net positive or negative impact on jobs. Instead, the goal was helping students to better understand outsourcing, techniques for dealing with outsourcing, and the limitations of outsourcing.

Despite the attention given to outsourcing there is not an existing body of knowledge about how to incorporate it into the academic curriculum. Some have reported course experiences with distributed or global software engineering [7]; however, these courses were not structured as outsourcing.

In general, the goal of introducing experience with outsourcing into computer science education can be seen as part of the trend to

bring academic preparation more in line with the needs of real world software practitioners [2] and to refocus education on the necessary skills for success in contemporary computing careers [1].

The popular press of the field has begun to focus on the perception that colleges and universities are not appropriately preparing future employees for computing jobs [3], including the demands of working with outsourcing. Adding outsourcing experiences into education can counter this perception.

3. CLASS STRUCTURE

3.1 Course Content

Both authors had previously taught an advanced software engineering course. Such a course is taken by undergraduates after they have considerable programming experience and possibly some small team project development experience. The typical student is a junior or senior computer science major.

The software engineering syllabi at both universities were similar - covering project planning, requirements, estimation, design, implementation, testing, and documentation. The overall course goal is to provide students a first hand experience using the techniques of software engineering in a large, semester long, team project.

3.2 Outsource Experience

Desiring to incorporate outsourcing into the course, the authors needed to decide how the two separate classes would be linked together with outsourcing. It seemed undesirable to have one university's class be the project lead and the other the site of the outsourcing – students would undoubtedly perceive that one role was more important than the other.

Instead, the authors wished to have each team on an equal footing – seeing the other location as equally important as their own. Hence, the selected organization structure was for teams at both universities to have the full experience: each team would run their own development and use a team at the other university as an outsourcing center.

The teams at each university were 4 to 5 undergraduate students; teams were formed by the authors based on a semester start survey of student past experience both inside and outside the class room. All teams programmed using Java and a total of 40 students were in the classes.

3.3 Inter-Campus Structure

The goal was to have the students experience a true, real world outsourcing experience – one similar to two companies or organizations working together in the global economy. In the real world, outsourcing does not take place between two groups that are completely unknown to each other.

Hence, it was necessary, in the limited time of the academic semester, to create a relationship between the two locations. First, the authors took still photographs of the individual students and showed them to the students at the other locations. Students were given online access to these photos to help them get to know the members of their paired remote team.

Second, voice-only teleconferences were used for brief sessions for the two locations to interact. These interactions were not technical content but mainly social.

As the third and final mechanism established by the faculty, each team identified a primary contact point for the other team to use in team communications. The primary contact was responsible for coordinating communications with the other team

Beyond this structure, the students were given freedom to define and use other forms of communication. In class sessions, the students were introduced to basic interpersonal communication concepts (e.g. synchronous and asynchronous communication, meetings with agendas, web discussion boards). See section 4.3 for a description of the inter-team communication mechanisms that developed as the teams worked.

3.4 System Development Projects

The authors gave the teams two choices for projects, both game programs. The teams were to develop a complete two person game for either chess or go. The ability to play "against the computer" was required. Considerable freedom was given on user interface selection and the mechanism for computer move selection.

Chess was the more common choice by the teams. Faculty paired teams with a team at the other university doing the same game.

Each team prepared (and was graded based upon) the deliverables shown in Table 1. The authors provided templates for each deliverable and the class room sessions included extensive coverage of the related software engineering topics.

Table 1. Team deliverables

Deliverable	Description	Typical Size
Project Management Plan	Project schedule, project description, people resources and roles assigned, risk analysis	5 pages
Requirement Analysis Document	Use case model and natural language system requirements	6 pages
System Design Document	Object orient system design using UML and a general system description	3 pages
Outsourcing Agreement	"Contract" for deliverables and schedule between two teams	2 pages
Test Plan	Numbered set of planned tests with pre and post conditions	5 pages
User Manual	Installation and user guide	4 pages

3.5 Creating the Outsourcing Agreement

Except for the Outsourcing Agreement the other team project deliverables are not described in this paper (and are similar to typical software engineering courses). The Outsourcing Agreement was added to give structure to the workflow between the teams at the two locations.

The teams were required to outsource one-third of their system to the other location. "One-third" was defined to mean approximately one-third of the classes and methods in the design.

This size requirement was used to ensure that a major dependency was created between the two teams (so that a team could not just throw away the work of the other team and do it by themselves).

The team receiving the work was called the "remote team" – its assignment was done for the "main team". Key items in the Outsourcing Agreement were:

- Date for main team to send System Design Document to remote team and indicate outsourced items
- Date for remote team to return agreed upon software to main team
- Description of testing to be done by both main and remote teams (documented in standard Test Plan document)
- Acceptance criteria for outsourced code

Teams were required to keep all documents under configuration control and to ensure that the same version of the System Design Document and Test Plan were available to both teams at all times.

The teams were encouraged to view the Outsourcing Agreement as a negotiated document between both teams – not as a "demand" from the main team to the remote team. Prior to the formal submission to the instructors, both teams were required to approve the document.

4. FINDINGS AND LEARNING OUTCOMES

The courses at both universities were conducted as an experiment to determine academic effectiveness, the impact on software engineering learning, and the changes in student perception about outsourcing. Results indicate that the course was effective and students generally enjoyed the unusual experience. This section presents selected results from the course including both student self evaluations and the author's findings.

4.1 Changed Perceptions of Outsourcing

At the start of the course, all the student participants had existing impressions about outsourcing in general and most expressed an opinion on these four points in a semester start survey (only a small number replied "no opinion":

- Outsourcing is primarily a way for an organization to save money by developing software using cheaper employees
- Outsourcing is likely to cause fewer jobs for software engineers in the USA
- Outsourcing is a way for an organization to draw on skills that are needed but not present in the organization
- Outsourcing is a good way to speed up software development and get projects done on schedule

The same statements were evaluated by the students again at the conclusion of the course, both times using the same scale.

Interesting changes in the student's perception were seen on the first and third item. At the semester start, fully 92% of the students agreed or strongly agreed with the statement that outsourcing was primarily a way to save money during software development (Figure 1).

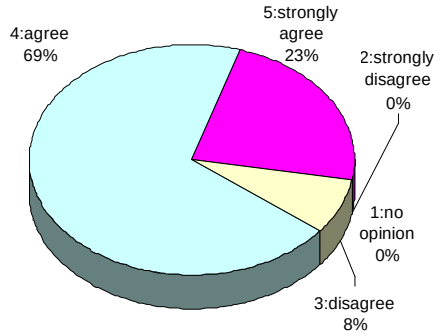

Figure 1. Outsourcing primarily to save money (course start)

During the project implementations the teams had considerable experiences with the reality of outsourcing. They learned more about the costs of outsourcing, particularly the need to have very complete design specifications and the time needed to communicate with the remote team.

All the teams reached agreements on their Outsourcing Agreement (see section 3.5) with limited difficulties. Typically the teams communicated openly with each other and came to an understanding quickly on what they would do for each other.

However, these agreements, possibly hastened by the deadline for turning in the deliverable for grading, were not completely effective in defining the work. The teams typically had to engage in considerable ad hoc communication about exactly what was to be done.

The authors encouraged the remote teams to return preliminary versions of their software to the main team; this step allowed the main team to check on the remote team's understanding of the design and requirements for the outsourced software. Typically, considerable mismatches were found and corrective actions taken. In some cases the main team decided to change their software rather than asking the remove team to correct their work. Often these mismatches were due to imprecision in the System Design Document (likely contributing to greater understanding of the importance of precise Object Oriented Design – see Section 4.2).

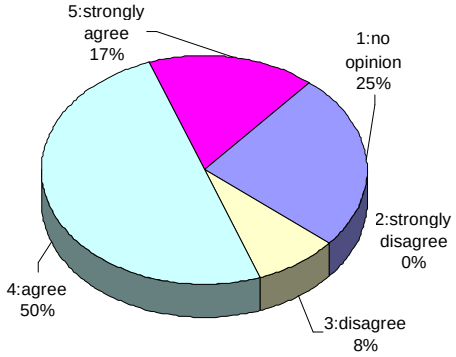

Figure 2. Outsourcing primarily to save money (course end)

As a result of the amount of work all the teams did to ensure a successful product from the remote team, opinions of the cost savings from outsourcing changed by the end of the semester. Only two thirds of the students still saw outsourcing being motivated by cost with a quarter of the students no longer being able to express an opinion on the topic (Figure 2).

These results may be influenced by the team's lack of experience with outsourcing. It is possible that with more practice they would again see outsourcing as a way to reduce costs.

However, one of the other items is unlikely to change with further experience – one that is a positive change in their perceived value of outsourcing. At the semester start, the students were about equally divided on the use of outsourcing as a way to draw on skills from outside the organization: 45% agreed or strongly agreed while 46% disagreed or strongly disagreed (see Figure 3).

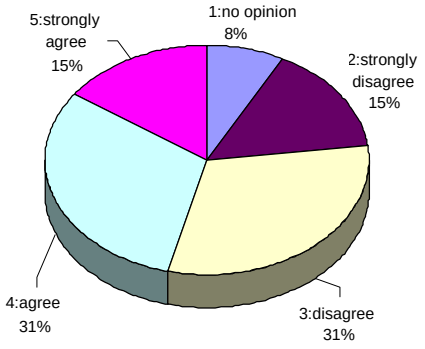

Figure 3. Outsourcing to add skills (course start)

During the team's work on projects, they became aware of the different skills on the team they were paired with. Since the course did not teach or mandate specific programming technologies, they were allowed to draw upon things learned in past classes. By including multiple universities there was a wider set of skills available for the team to draw upon. As a result the semester end survey showed all the students now either agreed or strongly agreed that outsourcing was valuable to add skills to an organization (Figure 4).

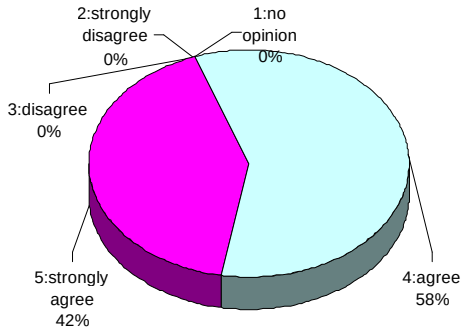

Figure 4. Outsourcing to add skills (course end)

4.2 Software Engineering Outcomes

Although outsourcing was an added emphasis in the course, the formal purpose of the course at both universities was to teach undergraduates software engineering. It was important that the added emphasis on outsourcing did not overly detract from that purpose.

Both student self evaluations and assessments by faculty were conducted on the learning outcomes from the course. In general, the course was successful in meeting its intended purpose. Further, the addition of the outsourcing experience seemed to increase the student's motivation in the course (which had sometimes been viewed as a difficult and somewhat boring topic by past students).

The semester start survey asked students to rate their personal knowledge levels in a number of software engineering technical areas. At the end of the semester, the same areas were again evaluated by the students as part of the formal class evaluation process.

Table 2 summarizes the student self evaluation and the changes during the class. In the self evaluation students rated their knowledge level using the scale 1: none, 2: a little, 3: medium, 4: a lot, 5: near complete.

Table 2. Software engineering self assessment

Knowledge Area	Average self rating - Start	Average self rating - End
Software Development Life Cycle and Software Process in General	1.9	4.0
Project Management, Scheduling, and Project Planning	3.0	4.0
Software Size and Cost Estimation	2.0	2.5
Software Requirements Creation and Validation	2.6	3.8
Object Oriented Design, including UML	2.5	4.5
Software Testing	3.2	3.8
Configuration Management and Change Control	1.7	4.1

As noted in section 3.1 above, students in the class have considerable software development experience from prior class work; many of the students have also worked in software development positions. Although it is a generalization, many feel they are already great software developers and have little to learn from a course some of them view as "management things".

The students have experience using object oriented design and UML from earlier courses although it tends to be limited to small examples. Hence, the large improvement in Object Oriented Design knowledge is likely due to their use of the tools in a larger

project and the fact that they better understand how critical it is to effective team implementation. The use of the System Design Document as a key part of the outsourcing contract increased the importance and utility of a complete design.

Configuration Management is not typically addressed in other courses, although some students encounter it in jobs. In general, starting knowledge of the topic was very low. Again, the need to communicate with the remote team likely heightened the need for careful configuration management and ensured that students learned to apply the key concepts.

Detailed discussion of the other software engineering knowledge areas and learning outcomes from the course are beyond the scope of this paper. However, the overall results were viewed as excellent by the authors and others at their universities.

4.3 Discoveries in Communications

As detailed in section 3.3, each team had a primary contact for communications with their paired team at the other university. Initially the team contacts exchanged email addresses and used email as their primary interactions.

Other studies have detailed the critical importance of communication for successful outsourcing [6], and the value of maintaining a good relationship and trust between the parties [8]. Hence, it is not surprising that the students quickly realized the value of good communications and begin to innovate using the typical tools of their generation.

In addition to using email for their deliverables and to exchange drafts, the student teams also communicated between universities in the following ways:

- Voice communications using both ad hoc and scheduled phone calls – these calls were usually two party calls and took advantage of mobile phone tariffs that did not charge extra for long distance calls.
- Text messaging using mobile phones – often used for confirmations that emails had been seen and to confirm when new information would be ready for the paired team.
- Instant messaging – teams added their counterparts both on the remote team as well as their main team to their IM "friends" and used their tele-presence in ways similar to both of the above.

Thus, the students adapted and used the same communications tools they used regularly in their personal lives. These additional tools were very useful in establishing and maintaining the positive working relationships between the teams.

5. CONCLUSIONS AND NEXT STEPS

Today, students (and the general public) have many perceptions about outsourcing, both in general and for software development. The course concept described here was created to bring the topic more fully into the academic teaching environment by providing students an outsourcing experience in the classroom.

The course, although taught only once to date, was successful both in engaging students in thoughtful exploration of outsourcing as well as teaching more traditional parts of software engineering. However, to be a broader and more inclusive experience the course concept can be expanded in these ways:

- Include teams from widely different time zones (both universities in the paper were in the same time zone) – the issues of "time shifting" put further strains on communication mechanisms but are common in real world outsourcing.
- Include teams from different countries and with different cultural backgrounds – other work (e.g. [5], [4]) has shown the importance of culture and its impact on both design and communications; a full exposure to outsourcing needs to include these complexities.

Both the above ideas can best be tested by cooperation in courses and outsourcing between universities more widely distributed across the globe. By doing so the field will continue to engage students and teach them how to deal with, and thrive in, the future global software community.

6. REFERENCES

[1] Denning, P. Recentering computer science. *Comm. ACM*, 48,11 (Nov. 2005), 15-19.

[2] Fernandez, J, Garcia, M., Camacho, D., Evans, A. Software engineering industry experience: the key to success. *Journal of Computing Sciences in Colleges*, 21,4 (April 2006), 230-236.

[3] Hoffman, T. Preparing generation Z. *ComputerWorld* (August 25, 2003)

[4] Krishna, S., Sahay, S., and Walsham, G. Managing cross-cultural issues in global outsourcing. *Comm. ACM*, 47, 4 (April 2004), 62-66.

[5] MacGregor, E., Hsieh, Y., and Kruchten, P. Cultural patterns in software process mishaps: incidents in global projects. *International Conference on Software Engineering*, (St. Louis, MO, USA 2005), 1-5.

[6] Oza, N., Hall, T., Rainer, A., and Grey, S. Critical factors in software outsourcing: a pilot study. *Proceedings of the 2004 ACM workshop on interdisciplinary software engineering research* (Newport Beach, CA, USA, 2004), 67-41.

[7] Petkovic, D., Thompson, G., and Todtenhoefer, R. Teaching practical software engineering and global software engineering: evaluation and comparison. *Proceedings of the 11th annual SIGCSE conference on innovation and technology in computer science education*, (Balogna, Italy 2006), 294-298.

[8] Taylor, H. Critical risks in outsourced IT projects: the intractable and the unforeseen. *Comm. ACM*, 49,11 (Nov. 2006), 75-79.