Report from the 2nd International Workshop on Software Engineering Course Projects (SWECP 2005)

Scott Tilley
Department of Computer Sciences
Florida Institute of Technology
stilley@cs.fit.edu

Shihong Huang
Computer Science & Engineering
Florida Atlantic University
shihong@cse.fau.edu

Kenny Wong
Department of Computing Science
University of Alberta
kenw@cs.ualberta.ca

Spencer Smith
Department of Computing and Software
McMaster University
smiths@mcmaster.ca

Abstract

This paper reports on the activities and results from the 2nd International Workshop on Software Engineering Course Projects (SWECP 2005), which was held on October 18, 2005 in Toronto, Canada. Creating software engineering course projects for undergraduate students is a challenging task. The instructor must carefully balance the conflicting goals of academic rigor and industrial relevance. Some of the fundamental characteristics of software engineering projects (e.g., team-based, large-scale, long-lived) are difficult to realize within the constraints of a university course in a single semester. This is particularly true when dealing with young students who may lack the real-world experience needed to appreciate some of the more subtle aspects of software engineering. This workshop explored how educators and industry can work together to develop a more rewarding educational experience for all stakeholders involved. Several key themes emerged from the workshop, including the importance of forming teams that are fair and balanced, the challenges in selecting a project that engages the students and meets the goals of the course, and the need for knowledge transfer amongst instructors.

Keywords: software engineering education, course projects, undergraduate students

1. Introduction

Team-based projects are the cornerstone of many undergraduate software engineering courses. In these projects, the students learn the importance of topics such as project management and issues of scale that separate software engineering from program development. For many students, the course project is their first exposure to working in a cooperative/competitive environment – quite different from the "start hacking, work alone, and don't talk to your friends" model that is typical of other software courses.

However, conducting such a course project can be a very challenging task. The instructor must carefully balance the sometimes conflicting goals of academic rigor and industrial relevance. Some of the fundamental characteristics of software engineering projects (e.g., team-based, large-scale, long-lived) are difficult to realize within the constraints of a university course in a single semester. This is particularly true when dealing with young students who may lack the real-world experience needed to appreciate some of the more subtle aspects of software engineering.

As an example, a typical undergraduate "Introduction to Software Engineering" overview course commonly that is offered at many universities is constrained by the length of the course (usually 10-15 weeks) and the typical challenges of attempting to recreate a commercial software engineering experience in an academic setting. Nevertheless, the students are subjected to some of the same challenges that their professional counterparts often encounter, for example, working on a tight schedule, with a team that is not of their choosing, and where their reward (their grade, in this case) is not entirely in their own hands.

From an industry perspective, many employers often lament that they must provide extensive (re)training to new employees. One of the reasons given for this situation is that the students haven't learned in school what the company considers to be important. To be sure, there will always be issues specific to the corporation that the new employee must acquire. But for software engineering, it seems odd that the projects students are given during their final years as an undergraduate are usually not indicative of the type of projects they will likely be working on in an industrial setting.

This paper reports on the activities and results from the *2nd International Workshop on Software Engineering Course Projects* (SWECP 2005), which was held on October 18, 2005 in Toronto, Canada. The SWECP series of events [3] was created to address some of the issues described above. The 2005 workshop explored how educators and industry can work together to develop a more rewarding educational experience for all stakeholders involved. Several key themes emerged from the workshop, including the importance of forming teams that are fair and balanced, the challenges in selecting a project that engages the students and meets the goals of the course, and the need for knowledge transfer amongst instructors.

The next section of the paper describes the workshop background, including goals, structures, and participants. Section 3 provides an summary of the workshop program, highlighting the main conversation points resulting from the presentations. Section 4 discusses several of the key themes that emerged from the workshop. Finally, Section 5 summarizes the workshop's results and outlines possible avenues for future work.

2. Workshop background

The SWECP 2005 workshop was a sequel to the workshop "Undergraduate Software Engineering Course Projects" that was successfully held as part of the CASCON 2002 conference [4]. CASCON [1] enjoys an enviable reputation as an event that brings together the top academics in Canada (and abroad), along with representatives of leading information technology companies and government agencies. The CASCON workshops in the past have proven to be very valuable to all involved. Two of the organizers of the SWECP workshop have participated in numerous CASCON events, dating back to 1991.

SWECP 2005 was actually the result of merging two workshop proposals together. The first proposal centered on software engineering course projects in the most general sense. The second proposal focused more on senior design/capstone projects. Putting these two workshops together into a single entity made good sense, and helped to attract a broader audience. The result was a stronger event with a more attractive program discussing a wide range of workshop goals.

2.1. Workshop goals

The primary goal of the SWECP 2005 workshop was to provide a venue for the exchange of practical ideas and information in the following areas:

- Selecting a course focus (e.g., analysis versus synthesis)
- Selecting a course topic
- The use of open source projects in an academic setting
- The use of commercial tools in the course project
- Involving local industry as proxy customers
- Quarter, semester, and multi-term length projects
- Managing multi-institutional course projects
- Dealing with individual grades for team efforts
- Monitoring and facilitating many teams in a term, yet providing timely feedback
- Increasing the appreciation of the softer aspects of software engineering (e.g., project management) in students who lack such experience
- Instructional technology support
- General issues related to software engineering degree programs, certification, and senior design/capstone projects
- Differences between undergraduate and graduate software engineering projects

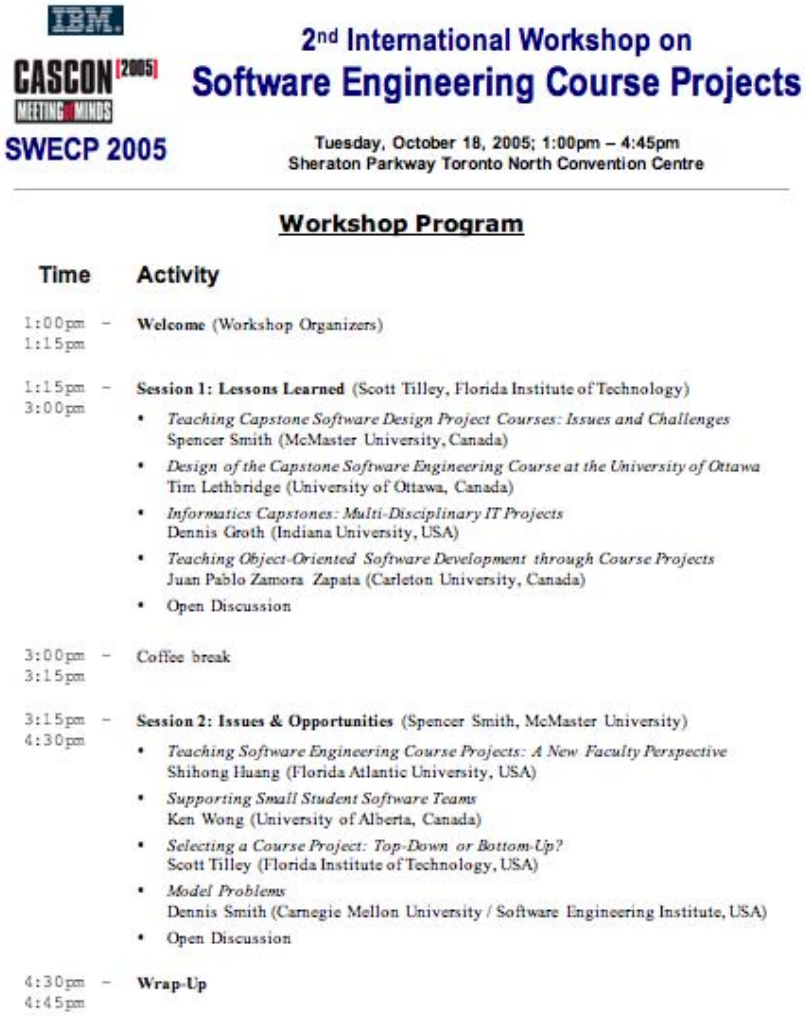
As with all CASCON workshops, a secondary goal is to foster collaboration among workshop organizers and participants. It is pleasing to note that this goal was also met; several post-workshop discussions arose that promise to develop into multiple-institute and inter-disciplinary endeavors.

2.2. Workshop structure

SWECP 2005 was a half-day event, held in a single room to accommodate the approximately 25 participants. As detailed in Section 3, the workshop was structured around theme-oriented presentations made by recognized experts in the field. Following each presentation, the workshop participants engaged in moderated discussion. The idea was to foster the exchange of ideas and information in an informal setting, but with some boundaries placed on topics and time to ensure that the workshop stayed on schedule.

2.3. Workshop participants

Since the workshop was part of CASCON, there was a good mix of participants from academia and industry. From academia, people most interested were educators responsible for teaching software engineering classes in a university setting. There were also a few graduate students who have their sights set on life as an academic after they complete their Ph.D..; they seemed to find the workshop extremely useful.



IBM CASCON 2005 MEETING OF MINDS

SWECP 2005

2nd International Workshop on

Software Engineering Course Projects

Tuesday, October 18, 2005; 1:00pm – 4:45pm
Sheraton Parkway Toronto North Convention Centre

Workshop Program

Time	Activity
1:00pm – 1:15pm	**Welcome** (Workshop Organizers)
1:15pm – 3:00pm	**Session 1: Lessons Learned** (Scott Tilley, Florida Institute of Technology) • *Teaching Capstone Software Design Project Courses: Issues and Challenges* Spencer Smith (McMaster University, Canada) • *Design of the Capstone Software Engineering Course at the University of Ottawa* Tim Lethbridge (University of Ottawa, Canada) • *Informatics Capstones: Multi-Disciplinary IT Projects* Dennis Groth (Indiana University, USA) • *Teaching Object-Oriented Software Development through Course Projects* Juan Pablo Zamora Zapata (Carleton University, Canada) • Open Discussion
3:00pm – 3:15pm	Coffee break
3:15pm – 4:30pm	**Session 2: Issues & Opportunities** (Spencer Smith, McMaster University) • *Teaching Software Engineering Course Projects: A New Faculty Perspective* Shihong Huang (Florida Atlantic University, USA) • *Supporting Small Student Software Teams* Ken Wong (University of Alberta, Canada) • *Selecting a Course Project: Top-Down or Bottom-Up?* Scott Tilley (Florida Institute of Technology, USA) • *Model Problems* Dennis Smith (Carnegie Mellon University / Software Engineering Institute, USA) • Open Discussion
4:30pm – 4:45pm	**Wrap-Up**

Figure 1: SWECP 2005 workshop program

Participation from industrial representatives greatly improved the workshop's tenor. For example, they were able to offer advice on relevance and desired skill sets. Many companies offer their own specialized training sessions, sometimes using mock projects that are similar to the projects university students' experience. It is only through dialog between academia and industry that the improvements can be made to software engineering course projects in the undergraduate curriculum.

There was no requirement for any participant to be currently teaching a software engineering course; we certainly did not exclude anyone from sitting in on the workshop. The main requirement was an interest in improving the current situation by actively participating in the workshop program.

3. The workshop program

The SWECP 2005 workshop was held Tuesday, October 18, 2005 from 1:00pm – 4:45pm at the Sheraton Parkway Toronto North Convention Centre. A reprint of the workshop program is shown in Figure 1. The workshop was split into two sessions:

"Lessons Learned" and "Issues & Opportunities." Each session had a moderator to guide the discussions.

3.1. Session 1: Lessons Learned

The "Lessons Learned" session ran from 1:15pm – 3:00pm, with four presentations of approximately 20 minutes each. This allowed for the speakers to present their views in a more in-depth manner, yet still allowed for sufficient interaction with the rest of the workshop participants.

The first presentation was "Teaching Capstone Software Design Project Courses: Issues and Challenge" by Spencer Smith of McMaster University. The mission of a capstone design project is to provide students with an opportunity to integrate what they have learned in earlier courses, deepen their understanding of that material, extend their area of knowledge and apply their knowledge and skills in a realistic simulation of professional experience. This talk discussed some of the issues and challenges that are important for supervising an effective capstone design project. The speaker also described potential solutions to address these issues and challenges.

The second presentation was "Design of the Capstone Software Engineering Course at the University of Ottawa" by Tim Lethbridge of the University of Ottawa. The talk covered issues such as how to ensure that students do a project that is industrially relevant, how to ensure the course integrates all the knowledge the students have learned in a wide variety of courses, how to grade group projects in a fair way, and what kinds of projects the students work on. The speaker also proposed how some of these ideas can be extended into projects that span multiple academic years, which might lead to a more problem-based learning style of education for software engineers.

The third presentation was "Informatics Capstones: Multi-Disciplinary IT Projects" by Dennis Groth of Indiana University. This talk presented experience in developing a full-year capstone course for the School of Informatics, which broadly studies the application of information technology in particular problem domain areas, including scientific domains like biology or chemistry, but also non-scientific domains like music, fine arts, or business. The speaker described his experience with the development of a capstone course for undergraduate Informatics students. He focused on process-related activities they undertook to deal with the rapid increase in enrollment for their course, which has grown from 15 students to over 120 students in just 4 years. He also compared and contrasted the types of projects, the range of skills, and the outcomes for their course with what might be experienced in a software engineering course. This is particularly important, since (as they have defined it) Informatics and Computer Science are distinctly different.

The fourth presentation was "Teaching Object-Oriented Software Development through Course Projects" by Juan Pablo Zamora Zapata of Carleton University. The presentation began with an outline of some of the continuing challenges for educators is finding a balance between theory and practice in the courses they teach. Although practice in undergraduate software engineering courses is key for the learning process, challenges like time constraints (e.g., the duration of the academic semester) substantially increase the difficulty of the task. Often courses focus on only a few aspects of software development leaving aside some others, resulting in an incomplete experience for the students. The speaker described how, for the last several years, the School of Computer Science at Carleton University has taught a software engineering course that incorporates each significant phase of software development in a course project, from requirements gathering

through implementation and testing. The students develop fully functional distributed systems in a Model Driven Development (MDD) environment using IBM's Rational Rose RealTime. The talk addressed some of the key elements of such course projects, as well how students manage to succeed in their implementation.

3.2. Session 2: Issues & Opportunities

The "Issues & Opportunities" session ran from 3:15pm – 4:30pm, with four presentation of approximately 10 minutes each. These presentations were shorter on purpose than those in Session 1, in order for the workshop to adopt a more interactive atmosphere for the second half of the event.

The first presentation was "Teaching Software Engineering Course Projects: A New Faculty Perspective" by Shihong Huang of Florida Atlantic University. The speaker began by describing how course projects are an integral part of undergraduate software engineering education, where students will practice techniques and theory they have learned in the classroom, and work in teams to plan, design, develop, and deploy software systems that follow software engineering standards. Since the speaker also participated in the first SWECP workshop when she as a graduate student, her insight as a relatively new faculty member into the different perspectives of the stakeholders involved in course projects was very enlightening. For example, the presentation emphasized the need for instructors to work closely with undergraduate students to provide detailed step-by-step guidance on how each milestones and deliverables should be finished, what kinds of documentation are expected, in what format, and so on. The speaker also said it was important to strengthen the students' abilities to learn on their own, but providing examples to learn by could be more beneficial to students just beginning to become full-fledged software engineers.

The second presentation was "Supporting Small Student Software Teams" by Kenny Wong of the University of Alberta. A central theme of this talk was that the success of a software development project depends on the technical competency of the development team, the quality of the tools they use, and the project-management decisions they make during the software lifecycle. Instructors of software-engineering courses that involve small project teams are often overwhelmed with the task of monitoring the progress of multiple teams. Without adequate monitoring and advice on best practices, problems in the teams process or the developed product may go unnoticed until it is too late to be easily fixed. The speaker outlined the JRefleX environment, with components built upon Eclipse, Wiki, and CVS, to support the education of small software teams. The environment has evolved over two years of use in an undergraduate user interface and software design course at the University of Alberta. Possible use of the environment for pedagogical research was also described.

The third presentation, "Selecting a Course Project: Top-Down or Bottom-Up?" by Scott Tilley of the Florida Institute of Technology, was removed from the program in order to allow more time for discussion. Fortunately, the main issues raised by this talk were covered during other presentations earlier in the day, and in the open discussion periods at the end of each session. A particularly important question was whether or not the instructor should impose a course project on the students (in a top-down manner), or if the students propose their own course project (in a bottom-up manner). Each choice has benefits and shortcomings; the tradeoffs between each of these options formed part of one of the key themes of the workshops (as discussed in Section 4 below).

The fourth presentation was "Model Problems" by Dennis Smith of Carnegie Mellon University's Software Engineering Institute (SEI). As part of the ISIS (Integration of Software-Intensive Systems) project [2], members of the SEI have embarked on a series of case studies to evaluate the efficacy of emerging technologies. The case studies revolve around "model problems": scaled-down case studies that focus on the essence of the problem, while still keeping the key elements of the application domain intact. For example, the speaker described a recent model problem involving the use of Web services as an integration mechanism for large-scale enterprise systems. Of particular interest to workshop participants was the invitation to involve graduate students in the ISIS project, by using model problems as the topic of M.Sc. theses and Ph.D. dissertations. This partnership shows significant promise for software engineering course projects as well. Indeed, not all course projects need to involve an intense software construction exercise; there is value in conducting sound analytical studies.

4. Key themes from the workshop

Several key themes emerged from the workshop's presentations and discussions. Three of the most interesting were the importance of forming teams that are fair and balanced, the challenges in selecting a project that engages the students and meets the goals of the course, and the need for knowledge transfer amongst instructors.

4.1. Forming teams

At the core of software engineering course projects is the tenet that the projects are executed by students working in teams. Beyond this agreed-upon fact, opinions diverged on most other aspects of the issue. Specific considerations include how teams are formed and members are selected (e.g., top-down versus bottom-up), how to fairly evaluate individual performance in a team context, and how to manage inter-personal dynamics and conflicts that commonly arise for students who are often working in a team setting for the first time in their undergraduate studies.

The latter point (students working in teams for the first time) is particularly important. When it comes to assigning roles and responsibilities within a team, sometimes a leader emerges without any external influence. In other cases, team members may benefit from trying out different roles during the project. For multi-semester projects, such role rotation can prove extremely beneficial. For multi-disciplinary teams, where some members excel at technical tasks while others are better communicators, such role rotation may not be as beneficial.

4.2. Selecting projects

At the very heart of software engineering course projects are the projects themselves. As with forming teams, one issue is how to select the project topics: top-down (by the instructor) or bottom-up (by the student teams). There are also practical considerations related to the nature of the project. For example, is it sufficiently challenging to warrant the students' attention for the duration of the course? Or, is it too ambitious and overly-optimistic for completion within a semester?

There was some discussion related to comparative evaluation of relative effort for different teams working on different course topics. If team projects are essentially judged

against their project plan, and not in relation to other teams, then how does one reward a well-executed project with moderate goals against a project that is not a success (according to their stated goals) but that was inherently more difficult to execute?

4.3. Transferring knowledge

It is inevitable that as one instructor builds up expertise in teaching and managing software engineering course projects, others will be moving on to other tasks, while at the same time new educators are just entering the field. The need to provide a mechanism to facilitate the transfer of knowledge between experienced and novice instructors was recognized as an important goal for the software engineering education community.

There was also benefit seen from an international repository of "best practice" when it came to software engineering course projects. For example, a database of course projects, including deliverables, schedule guidelines, evaluation criteria, and so on. One issue raised during the discussion was the portability of such materials to different contexts. Differences between software engineering programs, courses, and prerequisite knowledge across institutions creates difficulties in reusing course projects. If such issues could be address, then over time this repository could develop into a very valuable resource for all stakeholders.

5. Summary

This paper reported on the activities and results from the 2nd International Workshop on Software Engineering Course Projects (SWECP 2005). Approximately 25 people attended the workshop, which featured eight presentations structured into two sessions. Each session also included lively moderated discussions.

Three key themes emerged from the workshop: (1) the importance of forming teams that are fair and balanced; (2) the challenges in selecting a project that engages the students and meets the goals of the course; and (3) the need for knowledge transfer amongst instructors. The first two themes directly address some of the core issues related to software engineering course projects, and are likely to remain topics of discussion for some time. The third theme is related to the continuity and transference of best practice related to managing the course from the instructor's point of view; it too is likely to be a perennial topic of discussion.

Addressing each of these themes was deemed as the logical beginning of a follow-on workshop to SWECP 2005. A variety of possible venues were discussed; the decision is expected to be made by the first half of 2006. The workshop organizers welcome feedback from readers on the content and structure on the next event in this series.

References

[1] CASCON (CAS Conference). IBM Center for Advanced Studies, IBM Corp. Online at https://www-927.ibm.com/ibm/cas/cascon_main/index.shtml. Last accessed October 27, 2005.

[2] *Integration of Software-Intensive Systems* (ISIS). Carnegie Mellon University, Software Engineering Institute. Online at http://www.sei.cmu.edu/isis/. Last accessed October 18, 2005.

[3] SWECP. Online at www.swecp.org. Last accessed January 27, 2006.

[4] Tilley, S.; Boldyreff, C.; and Wong, K. "Workshop on Undergraduate Software Engineering Course Projects" (SWECP 2002: October 3, 2002; Toronto, Canada). Held in conjunction with IBM CASCON 2002 (Sept. 30 – Oct. 3, 2002; Toronto, Canada).