Lessons Learned Managing Distributed Software Engineering Courses

Reid Holmes
School of Computer Science
University of Waterloo
rtholmes@uwaterloo.ca

Michelle Craig, Karen Reid
Department of Computer Science
University of Toronto
mcraig,reid@cs.toronto.edu

Eleni Stroulia
Department of Computing Science
University of Alberta
stroulia@ualberta.ca

ABSTRACT

We have run the Undergraduate Capstone Open Source Projects (UCOSP) program for ten terms over the past six years providing over 400 Canadian students from more than 30 schools the opportunity to be members of distributed software teams. UCOSP aims to provide students with real development experience enabling them to integrate lessons they have learned in the classroom with practical development experience while developing their technical communication skills. The UCOSP program has evolved over time as we have learned how to effectively manage a diverse set of students working on a large number of different projects. The goal of this paper is to provide an overview of the roles of the various stakeholders for distributed software engineering projects and the various lessons we have learned to make UCOSP an effective and positive learning experience.

Categories and Subject Descriptors

K.3.2 [**Computer and Information Science Education**]: Computer science education

General Terms

Human Factors

Keywords

Software engineering education, OSS, project-based courses

1. INTRODUCTION

Computer Science departments often use upper-year 'capstone' courses to expose undergraduates to the experience of developing large-scale systems, more complex than the ones typically involved in their regular courses (e.g., [4]). These capstone courses are often 'greenfield', that is, students form small groups and propose and build systems from scratch. Unfortunately, this format does not match software-development reality: all team members are students, the work is not constrained by real users, and the systems do not suffer from an accumulation of technical debt. An alternative approach is to integrate students into real development teams. While cooperative education programs have long taken this route, they require a great deal of management and funding; additionally, it is often impractical, even infeasible, to fit long work terms into all students' academic schedules.

UCOSP, the Undergraduate Capstone Open Source Project, aims to blend these two models to provide students with real-world development experience in a format that fits into a single term and does not require a large amount of administrative overhead. We have now been running UCOSP twice a year for five years. UCOSP is a national program; faculty at any accredited Canadian university can send students. UCOSP is administered by a steering committee comprised of five faculty members. Since its inception, more than 400 students from over 25 universities have participated.

During the ten times we have run the UCOSP program we have learned a great deal about how to effectively manage a diverse set of students, projects, mentors, and faculty. We have previously reported on the general UCOSP structure [7] In this paper, we capture key UCOSP design decisions that we have found help maximize student benefit from the UCOSP program without overly burdening the students, mentors, or the steering committee. We wish to codify our hard-won experiences with UCOSP as other schools are starting new programs elsewhere (e.g., [6, 2]).

These design decisions predominantly involve clearly setting the expectations for each of the stakeholders involved in the program and organizing the program in a way that does not require full-time administrative support. This includes the choice of open-source (OSS) projects, the mentors from these projects, the students themselves, the home-faculty liaisons who incorporate UCOSP into their own curriculae, and the steering committee who organize and oversee the program. The goal of this paper is to better enable other universities to start their own capstone projects integrating students into real-world OSS development teams.

2. LEARNING GOALS

UCOSP has three primary intended learning outcomes:

Realism : All UCOSP students work on real distributed OSS projects as full members of software development teams. Students use the same software development processes as regular team members and are provided with explicit mentorship from volunteer mentors from each project. The value of mentorship from professional software engineers has long been recognized [1].

ICSE '14, May 31 – June 7, 2014, Hyderabad, India

Integration : Since students are working on real projects, they have an opportunity to integrate and apply the skills they have learned in their other university courses in a real development setting. In this way, UCOSP can act as the 'active experimentation' element of Kolb's Experiential Learning Model [5].

Communication : The majority of UCOSP interaction takes place online; this forces students to develop and improve their technical communication skills in a real development setting. Burge et. al. note the difficulty students have adapting their technical communication skills to workplace situations [3].

3. PROGRAM ADMINISTRATION

Organizing UCOSP consists of five primary components that have evolved with our experience.

3.1 Program Configuration

Establishing a new UCOSP-like program requires some initial setup activities that are significantly easier in subsequent years. These include sourcing funding, recruiting and selecting open-source projects, and gaining the participation of multiple universities. Recruiting the initial OSS projects was a challenge because potential mentors were unsure of what to expect from students and had many questions that the inexperienced organizers couldn't necessarily answer. The program had no track record and mentors where justifiably nervous about making the required time investment. Similarly, recruiting the initial participation of universities was a challenge. Administrators were nervous about the academic integrity of the activities, the supervision of the students, and the logistical details of determining grades. Once a dozen universities were able to publicly vouch for the success of their participation, others were keen to join.

3.2 Term Configuration

Before each term begins, we confirm the participation of each project and update its online description and set the sprint date and location. Students provide their project preferences in an online registration form. Approximately one month before the term starts, they are assigned to projects based on their project preferences and skill sets. At this point, some teams begin to connect and establish meeting times for the upcoming term. We have found that encouraging early communication means that students arrive at the sprint with development environments set up and familiar with the project; this enables teams to immediately focus on team building and productive technical work.

3.3 Code Sprint

Each term UCOSP hosts a three-day code sprint for all students and mentors. By meeting their fellow students and mentors face-to-face, students seem to better understand the goals of the project, are able to overcome the many technical challenges associated with contributing effectively to real projects, and feel like members of the team. The sprint also gives mentors the opportunity to assess the skills of each individual student in order to help pair them with appropriately challenging tasks. The majority of our past student problems have been associated with students or mentors who have been unable to attend the sprint; for this reason, sprint participation is now mandatory for all team members.

3.4 Midterm

Project mentors provide feedback to students part way through the course. More specifically, for each student they provide a letter grade, his/her rank in the team, a brief description of tasks undertaken and contributions to date, and a paragraph describing the student's strengths and weaknesses. This feedback often provides impetus to help students more effectively allocate their time and effort over the rest of the term. In particular, those students who have not effectively communicated with their teammates and mentors, or have refrained from sharing their code because 'it's not ready', realize that transparency is an essential aspect of successful team work. The ranking, which is not communicated to the student, encourages mentors to think more carefully about the relative grade assignments; it is also used to verify that the grades reflect the relative student ranking.

3.5 Final Evaluation and Wrap-up

Students are required to finalize their project contributions by their last day of class. This restriction exists because many universities prohibit course work during exam periods so the students can focus on their exams. This also prevents students from 'buying time' from their project mentors because they are busy with other coursework. The numerical grades and qualitative feedback provided by the mentors are scrutinized by the steering committee and then, after some negotiation with the mentors occasionally resulting in small adjustments, are passed along to the home-faculty liaisons. These home-faculty are responsible for submitting the course grades at their institutions and have the ultimate say in the grade. Some home-faculty use the grade from the steering committee directly as the student's course grade but others use it only as one component of the final course grade.

4. UCOSP DESIGN DECISIONS

Six primary stakeholder groups are involved in the UCOSP program. For each we have described their tasks along with various tips and tricks we have learned over time that improve the overall experience. These descriptions are steering committee-heavy as our goal is to help others form a steering committee to organize the own UCOSP-like program.

4.1 Steering Committee

The steering committee (SC) is responsible for all organizational aspects of the program. Currently, comprised of five members from four different universities, the SC is able to coordinate approximately 50 students without requiring significant additional administrative support.

The code sprint is the only UCOSP activity that requires any direct funding; historically, industrial partners have paid for these sprints. This requires that the SC communicate annually with these partners.

Three months before each term, the SC holds an informal discussion to determine which projects will be offered in the next term. In general, UCOSP does not have much project turnover as mentors are keen to continue to work with the program; however, when new projects are brought into the program the SC is careful to ensure that the project mentors understand their roles and the required time commitment.

Two months before term start, the SC contacts the home faculty at each institution to start the student recruitment process. While home faculty can select any students they wish, participants must be upper-year, full-time students who

are not currently on a co-op work term. Students register approximately a month before the beginning of the term.

From each accepted student, the SC collects some basic background data about technical expertise and experience and also project selection preferences. The SC maintains online project descriptions so students can make informed decisions about their project preferences. [1]. The SC then manually assigns students to projects. During this assignment, the SC a) assigns 4-8 students to each project (although 5-6 seems to be optimal); b) ensures a diversity of schools and geographic regions are represented on each team; and c) ensures that female participants are paired up on their projects. While the geographic diversity increases effort required to effectively communicate, it also forces students to work on these skills to best accomplish their assigned tasks.

The largest task for the SC involves organizing and managing the code sprint. Sprint activities start on Friday morning and end mid-day on Sunday (enabling participants to miss only only one day of work/school.) Students spend the vast majority of the sprint time working in their groups, although the SC organizes some team-building exercises at the start of the first day to help improve the teams' social cohesiveness. Each team also provides a 5-minute status report to all sprint participants at the end of the first and last days. A SC member visits each team to read them the 'riot act'; that is, to explain that the primary way students get into trouble in UCOSP is that they fail to keep in contact with their mentors and participate fully in the program. The SC also holds a mentors-only meeting to provide mentors an overview of the assessment process and to give them an opportunity to coach each other on successful mentoring strategies.

Timing the sprint can be problematic and we have found that the weekend following the second week of lectures to be the best choice. Mentors have expressed the desire to hold the sprint as early as possible so that it serves as a kick-off event. Twice we have held it later in the term (once after 3 weeks and once after 5) and both times we found that teams were less-productive overall. While some students could make contributions before the face-to-face meetings, others floundered. Because of lead-time for travel booking and resistance to asking students to miss any days during the first week of lectures, we have felt that we couldn't hold the sprint before classes start or during the first week.

Logistically, the sprint generally requires enough physical space for 50 students and 10 mentors to work together simultaneously; students mostly bring their own machines but projects often appreciate having access to a few large displays or projectors. UCOSP pays for the student's travel and hotel, and for some group meals. Participants are responsible for their other expenses. Collecting travel receipts and managing reimbursements is the least pleasant of the SC tasks; this is complicated by the fact that most students have not had to deal with these processes before.

Finally, the SC also manages the midterm and final review process. While the committee does not assign grades, it reviews each piece of feedback to improve consistency across teams and project mentors and to ensure that each student receives a meaningful assessment of their efforts. Midterm feedback is sent to the students and their home faculty but final feedback is sent only to the home faculty who will distribute it to their own students; this is because some participating institutions have additional course requirements (for example giving a presentation or writing a written report) that contribute to the final grade.

During the term, the SC also acts as a buffer between the project mentors and the home faculty liaisons. This is primarily because the SC consists of a largely fixed membership with established relationships with the mentors while the schools the students represent vary widely from term-to-term. In the infrequent case that a student problem arises during the term, the SC facilitates discussion between the project mentors and the home faculty liaisons.

One of the most difficult tasks for the SC is saying 'no'. The limited number of spaces each term means that some faculty are unable to send all of their highly-qualified students, mentors are unable to have more students on their projects, and there is insufficient space in the program for additional projects. While the program could be expanded to address these shortcomings, expanding beyond 50 students would require full-time admin support which would greatly influence the lightweight nature of the UCOSP program.

4.2 Projects

Attracting projects is not a problem; projects recognize the value that committed upper-year students are able to provide. Although it may be the case that a mentor could write the same number of lines of code as the mentee in the time taken to do the mentoring, mentors value the opportunity to recruit new young talent to their projects.

We require that all UCOSP projects be open source; this ensures that students will be able to freely talk about the work they have done in UCOSP and to use their project contributions as a part of their resumes once they graduate. It also means that there are no intellectual property concerns. Projects must also maintain active issue trackers, use version control, and perform code reviews. Over time, most projects involved in the program have migrated to using public (GitHub) repositories making it even easier for students to share the work they did in UCOSP.

Each project must have an active development community (to enable students to learn from other more experienced developers), and an active user community (to motivate students to produce a usable product). Projects also need to be able to put forward at least one mentor who will lead the students throughout the term. The most effective projects also tag specific beginner-friendly issues in their issue tracker.

For examples of projects past and present see [7]

4.3 Mentors

More than any other group, the mentors differentiate UCOSP from other university courses. Mentors teach students how to integrate into a real development community. They guide the students in the following development process and provide technical mentorship. Students receive detailed feedback on their contributions to the project. This feedback can be on students' status reports, how they interact in meetings, how they help each other, and in terms of their formal code reviews and patches.

Every UCOSP student performs a unique set of development tasks during the term. The mentor and student work together before and during the sprint to define a set of tasks that are engaging to the student, achievable during the term, and valuable to the project. It is important that these goals are tailored for the individual skills of the student; being able

[1] `http://ucosp.ca/projects`

set individual goals for students with different knowledge and skills, and focusing on the student's actual progress rather than simply the final product, is a pedagogic advantage of the program. When assessing the students, mentors are asked for detailed rationale for their grades. While at the midterm this is typically only a sentence or two, by term's end most mentors can provide multiple paragraphs along with links to online discussions and status reports, issues in the bug tracker, patches, and code reviews.

To participate in UCOSP, mentors must agree to attend the sprint, hold weekly (or more frequent) online meetings with the students, and write midterm and final evaluations for each student. Most teams hold online meetings using Google Hangouts and/or IRC; mentors are encouraged to require that each student submit a written status report a day in advance to facilitate an effective use of meeting time.

The most effective mentors are personally interested in helping the students to grow as software engineers. By being invested in the educational aspects of the program, mentors are able to make decisions that best help the student rather than solely focusing on the overall project goals. Over the course of the program, several former UCOSP students have returned as highly-successful project mentors.

4.4 Students

To be successful in UCOSP, students must be self-motivated. Since grades are determined predominantly by their performance throughout the term (no UCOSP school provides a final exam), students must be continuously involved with their team for the duration of the course. Students, home faculty, and mentors all have a clear understanding that students will spend 10–12 hours per week on UCOSP activities. Although they only receive two explicit grades for their efforts (the midterm and final), it is still critical that students maintain a steady pace each week. We emphasize this in a face-to-face meeting during the sprint but we often see a few students who fail to appreciate this before their midterm feedback only to greatly improve their performance in the second half of the term. In these cases, their assessment is largely based on their improvement, rather than penalizing them for their initial performance. Since successful UCOSP students need to be self driven, we do not believe the program to be a good fit as a mandatory course.

Students organize their own travel to the sprint and are reimbursed to a maximum amount (based on the student's university location and the sprint location).

In addition to working with their mentors, students end up working heavily with each other. Since the students are taking part in a shared experience (concurrently ramping up on a highly technical project), they are often able to answer each other's questions in advance of their weekly meetings.

Anecdotally we have found that students join UCOSP expecting that they will primarily be learning new technologies and contributing to a real project; however, they are often surprised by how much they learned about the process of developing software. Students often reflect that some of the most important skills they learned involved determining how to effectively communicate, manage their time, perform code reviews, and figure out how and when to ask questions.

4.5 Faculty

The primary tasks of the home-faculty liaisons are to recruit and screen applicants to the program, to ensure that once accepted their students are putting forth the required effort, and to handle administrative details at their home institution including submitting a final grade.

Faculty play an important role in ensuring that only highly-qualified, self motivated students are enlisted in the program. The SC committee establishes per-university caps on student enrolment numbers. In recent terms, the larger universities were able to enrol up to five students while most smaller universities were limited to two or three students. The SC assumes that home faculty will contact and personally connect with any student who is struggling; if this happens, it has typically been the mentors who contact the SC with a developing problem and the SC contacts the home faculty to address the concern. While some home faculty take a hands-off approach to UCOSP, the SC has been strongly encouraging faculty to connect regularly (bi-weekly or weekly) with their students.

Most universities provide 'independent study' credit for UCOSP; the SC suggests that at least 80% of the final grade be attributed to the work they performed on their projects.

5. CONCLUSION

Designing effective real-world capstone projects is a difficult task. We have run the Undergraduate Capstone Open Source Project (UCOSP) capstone course for 10 terms involving over 400 students from more than 25 Canadian universities. The program itself runs much like an open source project where the contributors (mentors, faculty, and steering committee) share in the primary activities. This allows us to provide a rich learning experience for students without any full-time administrative or academic staff. We hope these guidelines will help other faculty design their own capstone projects that integrate students into real-world experiential learning environments.

6. REFERENCES

[1] Curriculum guidelines for undergraduate degree programs in software engineering. `http://sites.computer.org/ccse/SE2004Volume.pdf`, 2004.

[2] Learning software development – by developing software. `http://web.mit.edu/newsoffice/2013/6s194-developing-Soft.-0424.html`, 2013.

[3] J. E. Burge, P. V. Anderson, M. Carter, G. C. Gannod, and M. A. Vouk. Integrating communication instruction throughout computer science and software engineering curricula. 2011.

[4] R. F. Dugan. A survey of computer science capstone course literature. *Comp. Science Ed.*, 21(3):201–267, 2011.

[5] D. Kolb. *Experiential learning: Experience as the source of learning and development*. Prentice Hall, 1984.

[6] M. Nordio, R. Mitin, and B. Meyer. Advanced hands-on training for distributed and outsourced software engineering. pages 555–558, 2010.

[7] E. Stroulia, K. Bauer, M. Craig, K. Reid, and G. Wilson. Teaching distributed software engineering with UCOSP: The undergraduate capstone open-source project. pages 20–25, 2011.