UnLecture: Bridging the Gap between Computing Education and Software Engineering Practice

Vignesh Subbian, University of Cincinnati

Vignesh Subbian is an instructor/teaching assistant in the Department of Electrical Engineering and Computing Systems at the University of Cincinnati. His research interests include embedded computing systems, medical device design and development, point-of-care technologies for neurological care, and engineering education.

Dr. Carla C. Purdy, University of Cincinnati

Carla Purdy is an associate professor in the School of Electrical Engineering and Computing Systems, College of Engineering and Applied Science, at the University of Cincinnati and an affiliate faculty member in UC's Department of Women's, Gender, and Sexuality Studies. She received her Ph.D. in Mathematics from the University of Illinois in 1975 and her PhD. in Computer Science from Texas A&M University in 1986. She is the head of UC's B.S. in Computer Engineering Program and the coordinator of the Preparing Future Faculty in Engineering Program. Her research interests include embedded systems and VLSI, intelligent embedded systems, software and systems engineering, computational biology and synthetic biology, agent based modeling and simulation, mentoring, and diversity in science and engineering.

UnLecture: Bridging the Gap between Computing Education and Software Engineering Practice

Introduction

The University of Cincinnati (UC) is considered to be the birthplace of co-operative education (co-op), with UC celebrating the 100-year anniversary of cooperative education, locally referred to as "reality learning"[1], in the year 2006. The co-op program at UC requires students to alternate between taking academic classes and working in full-time professional job assignments. While co-op is optional for some programs, it is a mandatory requirement for all engineering programs at UC, which are specifically designed as 5-year programs to allow for students to complete the necessary co-op requirements[2, 3]. Undergraduate engineering students complete five co-op rotations between their sophomore and senior years, accumulating close to 20 months of professional work experience in their field of study. Although instructors often relate concepts presented in the classroom to students' cooperative education, there is a need for teaching methodologies to better integrate every student's own real-world experience into engineering classrooms. Our hypothesis is that reflecting on and disseminating knowledge and experience that students gain through professional practice, in the context of a specific course in the curriculum, can be a valuable resource for classroom instruction. Based on this hypothesis, we have developed a novel pedagogical strategy called *UnLecture* that uses concepts from active learning and peer instruction to fully integrate students' co-op experiences into their classroom activities. This technique can also be applied in courses where students have worked in internships.

UnLecture Overview

An *UnLecture* consists of a reflective writing component and a participant-driven discussion. Each *UnLecture* session is based on a theme directly related to one of the course topics. Typically, an *UnLecture* on a topic is scheduled after that topic has been covered in an in-class lecture. A rubric is provided to the students a few days prior to the session. The rubric is the central element facilitating various components of this technique, including the Student Learning Outcomes (SLOs). It is a set of carefully designed questions divided into three sections: Retrospection, Examination, and Reflection.

- Before the session, students *retrospect* their past co-op/internship assignments, recollect details that are related to the session theme, and document some fine points based on the questions in the rubric.
- During the session, students share their retrospective thoughts and learn from fellow students' cooperative education experiences. They also *examine* practices that were realized in various course projects and assignments and analyze the differences and similarities between their experiences in industry and their learning experience from the course.
- After the session, the students combine their perspectives from both retrospection and examination to *reflect* on how they will perform differently in their next co-op rotation or work assignment.

Five *UnLecture* sessions were designed and executed as a part of the course EECE 3093C– Software Engineering during the Summer 2013 semester. The following is the list of session themes:

1. Project Management and Team Work
2. Requirements Analysis, Design, and Modeling
3. Software Implementation Techniques and Practices
4. Testing and Code Maintenance
5. Ethics and Technology/Patent Wars

It can be observed that session themes are closely related to topics in the discipline of software engineering. Subsequent sections describe the rubric design and results related to each of the sessions and provide evidence of how the *UnLecture* technique helped in achieving course goals. The SLOs and the process model of the course are elaborated in a separate paper[5].

Student Demographics

The Summer 2013 class had an enrollment of ten students (5 computer science, 4 computer engineering, and 1 computer engineering technology). The class size was significantly lower when compared to historical enrollment data because several students were on a co-op rotation during the summer in order to account for the recent academic calendar conversion from a quarter to a semester system. The small class size, however, provided an opportunity to implement and carefully assess feasibility of *UnLectures*. Out of the 10 students, 8 students had completed at least 1 co-op rotation in the industry. Two students did not have prior co-op or industry experience because the student transferred from a different major or institution that didn't require co-op as a part of their degree requirements or the student chose to do an on-campus research co-op and has yet to pursue an industry co-op rotation. Each student without co-op experience is paired with an experienced peer for *UnLecture*-related activities. In this way, these students receive firsthand information on industry expectations and the nature of co-op.

***UnLecture* I: Project Management and Team Work**

This is the first of the five participatory sessions that were conducted during the semester. The primary objective of this session is to discuss different team models and project management aspects, and also to provide an opportunity for students to get acquainted with the format of the session. Table I shows the rubric for this *UnLecture*. In the retrospective part of this session, students provided background information related to their co-op position and then presented the organizational model and managerial aspects of the company/team that they worked for. Figure 1 shows the distribution of Summer 2013 students' employers based on company size. Figure 2 shows the distribution of students' roles in their co-op assignment. The size of specific teams within the company varied from 2 to approximately 30, with some teams distributed across the nation or in different countries. It can observed from both Figure 1 and Figure 2 that, for a class size of 10, there is a good distribution of employers, ranging from start-ups to well-established enterprises, and at least one student in each of the following roles: designer/architect, developer, test engineer, and application support engineer. This kind of diversity in co-op assignments is

extremely useful and important because students receive different perspectives on working in the software industry. In the next portion of the session (examination), students discussed their views on the pair programming model of the course in relation to the team model of their co-op assignment. After the session, students documented the last section of the rubric (reflection). The following are some excerpts from reflection:

- "All of my experience has been in very small teams and it was interesting to hear about teams that were 25+ people …and about teams that were international and the benefits and difficulties of having people working at different time zones across the world."
- "It [the course team model] is different from my internship, where I was stuck alone for a long time with frequent unclear instructions."
- "It was interesting to see how their [fellow students'] co-ops were different from mine, especially those who worked on testing teams. I hope to gain experience doing testing in this course that I will be able to take back to my next co-op."
- "On my next co-op term, I would like to work on a larger team rather than just two people."
- "…because I have not yet completed a co-op, I do not have a good idea of what a co-op entails. Listening to my classmates talk about [co-ops]… has given me more insight and confidence that material learned in this class will be relevant and useful for my first co-op."

Table I Rubric for *UnLecture* I: Project Management and Team Work

Objective: The purpose of this assignment is to document and share your project management and team work experiences from your co-op, internship, or other industry work experiences. Prelude: In this section, you are required to include the following information related to at least 2 past industry projects: title/role, company/organization (optional), and overview of the team that you were a part of, summary of your responsibilities, duration of each assigned task, and duration of the project.
Retrospection: For each project you listed, write a retrospective essay based on the following questions: 1.1. What was your team model (team size, who led the team? How were tasks delegated? etc.)? Describe the technical nature of the team (example: development or testing or both, support etc.) 1.2. Who made important decisions in the team? How were decisions made? 1.3. What kinds of meetings were conducted (example: stand-up meetings, all-hands weekly meetings, town-hall meetings, etc.) and how often? How were meetings/decisions documented and communicated? 1.4. What problems/issues did you encounter while working in teams? How did you resolve them? What were some benefits of team work?
Examination: 2.1. What are your thoughts on the team model of this course? Besides size, how is it different from or similar to your past experiences (include pros and cons)?
Reflection: 3.1. What did you learn from this *UnLecture* session? In other words, what are your thoughts on conversations that you had with your classmates and their experiences? 3.2. Based on this discussion and the team work experience in this class, what will you do differently in your next co-op or industry work assignment?

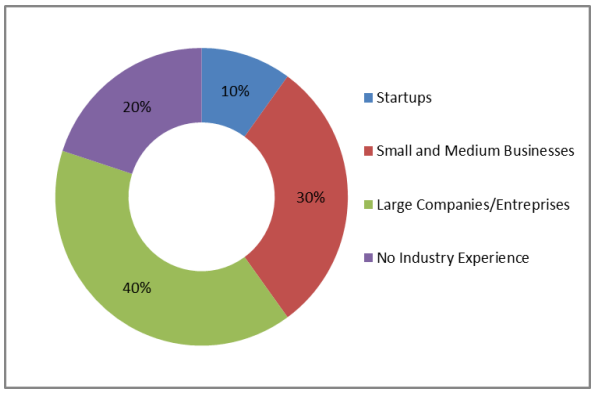

Figure 1 Distribution of Students' Industry Experience

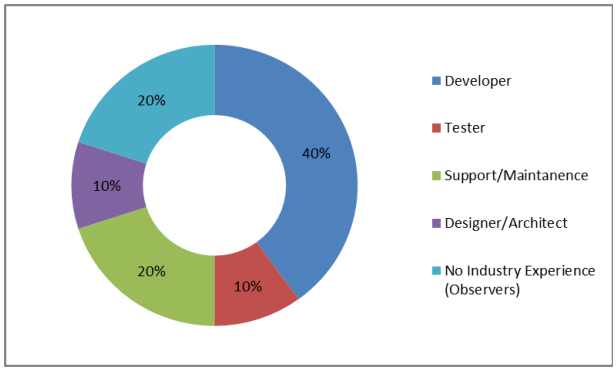

Figure 2 Distribution of Students' Role in Industry

UnLecture II: Requirements Analysis and Design

The focus of this *UnLecture* is to document and discuss software development life cycle (SDLC) models, business process models, requirements analysis, and design and modeling experiences from students' co-op assignments. Table II presents the rubric for this *UnLecture.*

Table II Rubric for *UnLecture* II: Requirements Analysis and Design

Retrospection: 1.1. What SDLC model did your team/company use? Who were customers to your team? Were they internal or external? How were requirements obtained from customers? How were requirements documented and communicated? How clear or unclear were the requirements? How did you/your team deal with unclear requirements? 1.2. What design strategies did your team use before starting implementation? What tools/techniques did you use for design/modeling? Did you have brainstorming (or "powwow") sessions? Elaborate.
Examination: 2.1. What are your thoughts on requirements analysis, design and modeling techniques used this course? How are they different from or similar to your past experiences?
Reflection: Similar to the *reflection* section in Table I

SDLC models are usually covered at the beginning of the semester, and this *UnLecture* was a good place to revisit and discuss those models, especially because it is also an important SLO of

the course[5]. The course uses a combination of waterfall and agile models for the laboratory project. Figure 3 shows the distribution of SDLC models that students used in their co-op assignment or previous software-related projects. An important objective of this session is to emphasize the connection between the software development process and the application needs. The following excerpts from student reports are examples of how *UnLecture* furthers the understanding of concepts that students learn from traditional lectures and laboratory projects.

- "I do remember seeing a diagram (in my co-op) that was made during one of our meetings … I believe it was a class diagram, since it showed what some classes would contain and what methods we would need to implement. I didn't know UML then…"
- "It does seem like it takes a lot of time to create models for a software project but it will force you to think and know how the software will be structured and designed. This can also lead to solving many issues that may arise before any coding is actually begun."
- "Another point from this Unlecture I found interesting was when one student talked about his experience as a free-lance web-site developer. As I've never been responsible for requirements gathering myself, I thought his perspective on customers on opposite ends of the spectrum was very interesting…"
- "When designing large systems, I can see how it (structured design) could be useful…"

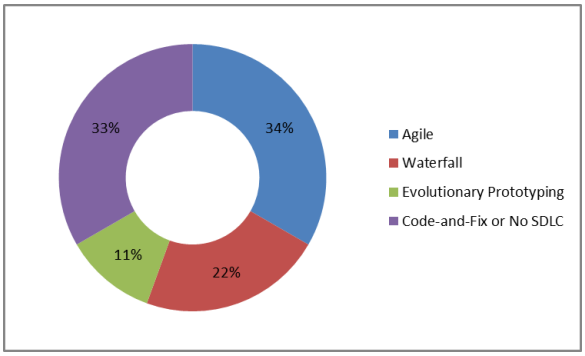

Figure 3 Distribution of SDLC models used by employers of Summer 2013 class

UnLecture III: Software Implementation Techniques and Practices

In this *UnLecture*, students discuss software construction and programming techniques, with specific focus on the object-oriented programming (OOP) paradigm. The rubric for this session is shown in Table III. The inherent goal of this session, as demonstrated in the rubric, is to reinforce the fundamentals of OOP and broaden students' ability to follow generally accepted programming practices. Students provided specific examples of their programming styles and how they applied their theoretical knowledge of OOP concepts to actual practice, which was evidently useful for students who generally follow a code-first-and-fix-later approach. It was intriguing to see how peer instruction successfully inserted itself into the process. The following are excerpts from both student reports and actual dialogue during the session.

- "(for this *UnLecture*), I also had to really think about, what and how I think about before I start to program. It seems like the majority of my programming has been code and fix approach."

- "A new concept that I learned while on my coop was the practice of defense in depth. Defense in depth means … (explains the concept with example)."

- "I realized I was attempting to do things in a procedural mindset instead of using OOP."

- "I often use *rubber ducking* for debugging, and sometimes even for designing my classes and methods … (explains how the technique works)."

Table III Rubric for *UnLecture* III: Software Implementation Techniques and Practices

Retrospection: 1.1. Briefly describe a software module that you were tasked with. Was it a part of a system/application or was it a part of an automation/scripting? In what language(s) did you program? 1.2. What OOP concepts did you use while programming at work? Give specific examples. What software-related concept(s) did you realize during the course of a specific project? (In other words, you knew the concept theoretically, but actually applied it while working on the project). What "best" programming practices did you follow/learn? 1.3. Explain your thought process during a typical programming session (This is an open-ended question)
Examination: 2.2. Explain, with examples, OOP concepts and design patterns that you have used in this class.
Reflection: Similar to the *reflection* section in Table I

UnLecture IV: Software Testing and Code Maintenance

In this session, students with software testing experience were identified prior to the session and were assigned as session moderators. This was primarily done to evaluate *UnLectures* for larger classes. Also, not many students are exposed to testing in their initial co-op rotations; the majority of them work as developers (see Figure 2). In this class, there were two students with shadowing experience in testing and quality assurance (QA) teams, and only one student who actually worked as a testing intern. These three students led the discussion based on the rubric shown in Table IV, and other students benefited from these leaders' experience in software testing. Additionally, test-first development is central to the design process of this course and hence, a significant amount of time was spent on discussing testing experiences from the course (see *examination* section in Table IV). This *UnLecture* also covered topics in code maintenance, as presented in the rubric in Table IV. Typically, lectures/*UnLectures* are an hour long, but given the breadth of topics in this *UnLecture*, and the need to discuss both testing and QA topics in the same session, a 90-minute meeting is recommended. The peer-moderated format of *UnLecture* was well-received by all students, and several of them produced very detailed and well-thought-out summary reports. A few excerpts from the reports are as follows:

- "…I also learned that it is good practice to write a test case whenever a bug is found, in case the bug ever resurfaces."

- "I thought that the way he (a fellow student/moderator) was able to come in with prepared examples of how testing was done within his company was very informative to someone like me who really has had no formal experience with software testing."

- "When I worked with the team running stress tests, it was actually not led by the testers, but by a lead developer on the team. It consisted of… (explains how stress tests are run on databases)." This is an example of a topic that was presented minimally in the lecture, but was re-visited and covered with examples in an *UnLecture.*

Table IV Rubric for *UnLecture* IV: Software Testing and Code Maintenance

Retrospection: Part I: Testing 1.1. If you were a part of a testing or QA team, explain in detail the testing practices of the team, test suite, kinds of tests that were written or used, quality of tests (i.e., ability of the tests to find bugs), and build/deployment process, and, finally, share your learning experience from the testing/QA team. 1.2. If you were not a part of a testing/QA team, explain your interactions with the testing team. 1.3. Besides just compiling/running, how did you test the code correctness? How did you ensure that you did not break the main build? 1.4. How did you debug your code? What are some common debugging practices/tools of your team? Give specific examples of bugs that you encountered. How did you fix them? Part II: Code Maintenance 1.5. Explain code maintenance practices of your team (give details about version control and associated tools)? How often did you update, commit and/or merge, and build? 1.6. Explain the release cycle of your team. How were releases made to customers? How often did customers receive releases? 1.7. When did you first learn about version control? After learning to do version control, how did it change your views on maintaining code?
Examination: 2.3. Share the kinds/levels of tests you wrote as a part of this course. Give specific examples, if any, of how tests found bugs in your code? 2.4. What did you learn from writing test cases and test plans? 2.5. What are your thoughts on code maintenance and release cycle in this course? How are these different from or similar to your past experiences (pros, cons)?
Reflection: Similar to the *reflection* section in Table I

UnLecture V: Software Engineering Ethics and Technology/Patent Wars

This is the last participatory session in the series of *UnLectures* that were conducted in Summer 2013. It consisted of one typical course topic on software engineering ethics and one non-traditional discussion topic on patent wars. This *UnLecture* is an example of adding an active learning component to abstract topics such as business and legal aspects of software engineering, along with information that doesn't necessarily require lecturing, but may require research or critical thinking, such as the IEEE/ACM Software Engineering Code of Ethics. Instead of listening to lectures, students use their research and reading to inform their discussions during *UnLectures.* The primary focus of this *UnLecture* is to discuss ethical issues and dilemmas related to software engineering in both industry and academia. As a research exercise, student

teams are also assigned readings on an ongoing or past technology/patent war in the software industry. This exercise is meant to broaden their awareness of software Intellectual Property (IP) and associated infringements. The rubric for this *UnLecture* is show in Table V.

Table V Rubric for *UnLecture* V: *Software Engineering Ethics and Technology/Patent Wars*

Retrospection: Part I: Ethics 1.1. What are your personal ethical principles related to a) workplace b) software engineering. You may give specific examples. 1.2. What ethical questions have arisen in your professional experience? Explain how you (or the person involved) resolved the dilemma? Relate each experience to a clause in the IEEE/ACM Software Engineering Code of Ethics (include the clause #). 1.3. Pick a specific clause from one of the 8 principles in the IEEE/ACM Software Engineering Code of Ethics (include the clause #). Critique the selected clause quantitatively. Include examples, as needed. Note: Avoid using the same clause for both (1.2) and (1.3). 1.4. Were you given any kind of orientation/training, formal or informal, on ethical practices, as a part of your co-op/internship? 1.5. Explain general work/business ethics of your team/company (example: policies regarding data storage, server access, access to internet content during work, work-from-home options, etc.) Part II: Technology Patent Wars: Research and investigate the patent/IP war that was assigned to you, and then answer the following questions: 1.6. Briefly describe the case and involved parties (who initiated the lawsuit (plaintiff) against whom (defendant), what was the plaintiff's claim? etc.) 1.7. Explain specific technical details (related to hardware, software, design, and/or name/logo) behind the claim/IP violation? 1.8. What was the outcome of the lawsuit? What is your take on the outcome (potential/favorable outcomes, if the suit is still ongoing)? Include your own perspectives. 1.9. (Open-ended question) In your own opinion, what is the next big thing in the software industry? Which technology has the potential to revolutionize the software industry?
Examination: 2.6. What ethical questions did you face in this course and how did you resolve the dilemma?
Reflection: Similar to reflection section in Table I

Results and Discussion

A total of 15% of the course grade was allocated to *UnLectures*. Attendance and participation in the five *UnLectures* constituted 5%, and the reflective writing components, before and after the sessions, were assigned the remaining 10% of the grade. The rubric is provided a week before the session to allow for students to complete the *retrospection* and *examination* sections. Students are advised to complete the *reflection* section on the same day as the session, and submit the entire report within a day after the session. *UnLecture* evaluation results including excerpts from student feedback are shown in Table VI. Besides minor criticism on the amount of writing an *UnLecture* entails, the student responses were mostly positive and appreciative. The writing component is extremely important for achieving SLOs and hence, it may not be combined into a group assignment. Finally, the success of *UnLectures* lies in careful placement of the sessions within the course calendar. The Summer 2013 semester was in fact a controlled

experiment to determine the placement of the five *UnLectures* in relation to the lectures and the release cycle of the course laboratory project[4].

Table VI Evaluation Results

Course-specific Question	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
The *UnLecture* sessions helped in relating my industry/work experience to this course (and vice versa)	0%	0%	20%	40%	40%
The combination of both in-class lectures and *UnLecture* sessions enhanced the learning experience in this course	0%	0%	0%	60%	40%
Excerpts from student feedback					
1.	"The unlectures were really fun. I really enjoyed talking about and hearing others' perspectives from industry."				
2.	"This was a very informative course and I learned a lot. I could relate a lot of what I did in co-op to this course."				
3.	"I thought that the unlecture sessions were a good addition to the course."				
4.	"These [questions/issues discussed in unlectures] also came up in several of my co-op interviews this year and I believe that the background information I learned in this class played a major role in some of the jobs I was offered."				
5.	"I didn't expect to write 5 essays for an engineering class."				

Conclusion

This paper has described a novel pedagogical strategy to methodically integrate outcomes from both experiential and classroom learning. The software engineering *UnLecture* rubrics presented in this paper are useful examples for understanding the technique and recreating them for other engineering disciplines. The rubrics can be tremendously valuable to software engineering educators, particularly to those in institutions that have integrated cooperative education or internships in their academic programs. With careful planning and rubric design, *UnLectures* can be integrated into other electrical and computer engineering courses[5] and also into courses in other engineering disciplines as well. Finally, it is worth mentioning that, besides providing evidence of how students value the *UnLecture* technique, students' experiential learning reports are reliable sources to inform teaching and cooperative education research. A detailed report on how the concept of *UnLecture* evolved and how it compares to other active learning techniques in the literature is presented in a separate paper[6].

References

1. N. L. Zimpher, and R. D. Wright, "100 Years of Reality Learning," *Community College Journal*, vol. 76, no. 3, 2006, pp. 24-26.

2. K. Cedercreutz and C. Cates, "Cooperative Education at the University of Cincinnati: A strategic asset in evolution," *Peer Review*, Fall 2010, vol. 12, no.4, pp. 20.

3. L. Harrisberger, R. Heydinger, J. Seeley, and M. Talburtt, *Experiential Learning in Engineering Education*, American Society for Engineering Education, 1976.

4. V. Subbian and F. R. Beyette "Developing a new advanced microcontrollers course as a part of embedded systems curriculum," *IEEE/ASEE Frontiers in Education Conference*, 2013, pp. 1462-1464.

5. V. Subbian and C. Purdy "A hybrid design methodology for an introductory software engineering course with integrated mobile application development," *ASEE Annual Conference*, 2014.

6. V. Subbian and C. Purdy "UnLecture: A novel active learning based pedagogical strategy for engineering courses," *ASEE Annual Conference*, 2014.