The Road to a Software Engineering Project Course: A Retrospection

Jane Chu Prey
Department of Computer Science
School of Engineering and Applied Sciences
Thornton Hall
University of Virginia
Charlottesville, VA 22903 USA
Email: prey@cs.virginia.edu

Abstract

Software engineering is acknowledged to be an important (and major) component in the education of today's computing professional. The way in which to present and incorporate this education into an undergraduate Computer Science curriculum is still being debated.

At the University of Virginia, we have undertaken to incorporate software engineering concepts and principles into all courses beginning with the very first. Our new curriculum was designed to be more mathematically rigorous, including more software engineering concepts and practice, more practice-oriented, more closely related to the real-world environment. Students are engaged in various aspects of software engineering - many of which appear in more than one course. This repetition of experience with software engineering concepts and practices provides students with solid grounding in software engineering and its importance in the computing profession.

Introduction

Software engineering is acknowledged to be an important (and major) component in the education of today's computing professional. The way in which to present and incorporate this education into an undergraduate Computer Science curriculum is still being debated.

At the University of Virginia, we have undertaken to incorporate software engineering concepts and principles into all courses beginning with the very first. Our new curriculum was designed to be more mathematically rigorous, including more software engineering concepts and practice, more practice-oriented, more closely related to the real-world environment.

The incorporation of these ideas into our teaching program lead to a need for an entirely new undergraduate curriculum. A number of substantial changes were made to our program, specifically:

- extensive revision of many existing courses. All of the core courses containing elements of programming, data structures, machine representation, etc. were completely redesigned.
- development of several new course, especially in discrete mathematics and computation theory.
- the replacement of the programming language used for teaching purposes. C++ was selected as the new programming language and although not ideal, it has the benefits of supporting object-oriented programming and increasing industrial acceptance.
- a belief in the importance of reuse of courseware. We intend to make our artifacts available to other universities that choose to adopt elements of our curriculum.
- the development of the "closed laboratory" facility to support closed-laboratory exercises. The facility is being expanded to include devices such as motorized vehicles and other elements that simulate realistic applications.

What and Where Software Engineering Happens in the Curriculum

We have just completed the first full cycle of the core courses in the new curriculum. Three of the four core courses were designed to contain a significant emphasis on software engineering.

The first two courses described below are the first two courses in the curriculum. The third course described is a software engineering project course. It is the fourth course of the core sequence.

What we makes our first courses (especially the laboratory component) of the curriculum revision different from other introductory Computer Science courses is our commitment to software engineering starting in the introductory courses. We have designed laboratory activities which explore specific software engineering concepts and practices.

Techniques and procedures for developing, maintaining and improving software products are not easily learned nor easily adapted to one's existing working approach. Many of these techniques and procedures can only be learned and understood by repeated expose and progressively more in-depth work. Students should be made aware early on that "real" software products can be hundreds of thousands or even millions of lines long. They should have some exposure to what "real" software is - how it happened, what makes it good (or bad), why it's important to have standards, how programmers and users interact with it, etc. We believe students can have meaningful software engineering experiences in introductory Computer Science courses.

As part of the development of the new curriculum, we defined a set of skills and knowledge and the appropriate level of learning for each course. The following sections discuss examples of the types of laboratory activities we have developed which incorporate the knowledge and skills defined for software engineering. The laboratory provides an opportunity for the students to experience the software engineering concepts and principles. A detailed table of knowledge and skills for software engineering for each course (as well as other course content specifics) can be found on our homepage - http://www.cs.virginia.edu.

Software Engineering in CS1

There are many software engineering techniques and procedures students need to be exposed to and have experience with before they can develop a true sense of the discipline. To begin building this framework, we have designed laboratory activities which allow the students to experience each of the knowledge and skills we defined. Some of the activities are short and straightforward while others are more involved, some are parts of several laboratories.

For example, students early in the semester have a laboratory activity which requires them to learn how to use the debugger. Later activities may require them to use the debugger. With repeated opportunities, the debugger as a tool for solving problems becomes an automatic response.

Another activity gives students an exposure to a large software system (80,000 lines of code). Before coming to lab, students respond to a questionnaire which asked about their perceptions of large software systems, the importance of design, documentation, etc.

Students spend the entire laboratory period (2 hours) working through a large software system. Students are first in the role of the user of the large software system - inputting information, using the various program options, seeing how it works.

Next, they are asked to work in teams of 2 or 3 to make what appears to be a minor change. This is not an easy task to complete. Very few groups successfully complete this portion of the lab. Most groups respond with frustration.

A discussion involving all groups in the lab follows focusing on what they just experienced - what strategies they used to try to solve the problem, what information was missing or incomplete (or misleading) that caused the task to be more difficult that they had expected, what else could they try to solve the problem, what should have been done to avoid the problems, etc. As a post laboratory assignment, students are asked to redo the questionnaire, compare these responses to their initial responses and to write a brief discussion on one of the questions, preferably one where their perception had changed.

Software Engineering in CS2

The laboratory activities again provide critical software engineering experiences in this second core course. The weekly lab meetings changed from single purpose sessions to 2- to 4-week laboratories where activities from one week to the next are connected, are more project/applications oriented.

For example, using a security camera mounted in the teaching lab, four consecutive laboratory sessions are designed to give students experience with software implementation, verification and maintenance. The first session is the first iteration in a two-step Spiral process.This laboratory introduces iterative software development, risk-based development and the Spiral software development processes, and the use of both real and simulated peripheral equipment. Students address the most important risk areas of this project by developing parts of the security camera software in prototype form.

The second lab session, the second iteration of the Spiral process, develops a real software application using information and prototype software from the previous lab. A Users' Manual is a required document.

The objective of the third lab is to perform a rigorous, formal inspection. The target of the inspection will be the source code for the security-camera application. In performing the inspection, every effort is made to locate defects of all varieties and to make the maximum possible improvement in the software product within the allotted laboratory period.

The fourth and final laboratory session of this sequence is used to test the security camera application.

This involves developing both a test harness and a test plan, and using the test plan on ones own software product. The goal is to run a completely automated test sequence with complete self-checking of the output on the security camera application software.

A Software Engineering Project in CS4

The fourth course in the core sequence is an advanced software development course. We offered it for the first time in Spring, 1995. It was designed to be a project course with the emphasis on producing a real product.

The project was to define, develop and implement the software required for a mobile robot control system. Students were given the scenario that there are certain environments where it is neither practical nor safe for a person to explore. The mobile robot can be used as a substitute person to explore these environments. Such environments include the surface of the moon, toxic waste dumps, and nuclear disaster sites.

The system controls the mobile robot from a fixed-station via a communications link. The mobile robot is capable of sending back a video signal, an audio signal, and telemetry information about the environment. Using a graphical user interface on the fixed station, a human operator can use the mobile robot to explore dangerous environments. The onboard software runs on the mobile robot, the fixed-station software on a pc, the communications link software runs on both the mobile robot and the fixed-station and acts as a link between the onboard software and the fixed-station software.

The 36 enrolled students were randomly divided into six 6-member teams to work together for the entire semester.

Second Thoughts

Although all six teams had a working software product at the end of the semester, things did not go as smoothly as we would have liked. We learned as many lessons as the students. Chief among them, we learned that having only one test board (which simulated the hardware on the mobile robot) for all six teams to use was not reasonable, that batteries are very likely to be out of power, that equipment in general is hard to keep in working order. If we expected the students to have a "real world" experience, we needed to provide a "real world" environment - they needed equipment that was dedicated to them (each team received a test board), working space (a room to meet in), storage space (file cabinets to store working documents, software, etc.), contemporary tools (MS Office, Powerpoint).

We also realized that this project was more sophisticated and complex that we had intended. Students were spending far too much time on this single course. One of the major problems we did not compensate for was the students' lack of in-depth experience with many of the aspects of a software development process. They were expected to understand and execute every facet of the software development life cycle from design to implementation, from interim project presentations to final documentation. We had not provide them with the necessary background for this level of expectation. We need to somehow scale back the project without losing the intent of the experience.

There are several approaches we are considering. Some of the options we have under consideration include dividing the course into a two semester using a similar project, giving more credit hours for the course and keeping it all in one semester, or developing a different course where students will look at specific parts of software development, learning them well with a project course the next semester.

It is interesting to note that although almost all of the students thought the workload was extremely high, responses to a survey given at the end of the semester indicated the students believed the course was very worthwhile and were willing to spend the time.

Student Responses

In addition to the normal end of the semester surveys, we were interested in the students' attitudes after they had some time to be away from the course. Six months after completing the course, students were asked for their thoughts on the experience. Typical responses were:

"For all the moaning and groaning our class made about the workload, I don't think a single person can say that they were not challenged by CS 340. I also don't think that a single person in the class can say that they didn't learn anything that they could use later in life. There are few other classes that I have encountered in my academic career for which these statements are true."

"To sum up: I thought CS340 was the most difficult class I have ever taken in my life, but given the chance I would do it all again simply because I have never learned so much."

"I just wanted to say that no matter what sinister thoughts I had during last semester in CS340, I am glad I did it because I would otherwise be screwed (now and when I enter the workforce for good)."

Summary & Conclusions

We have recognized the need for software engineering to become an integral part of a professional computer scientist's skills and knowledge. One course in software engineering taken late in one's academic career will not provide the needed foundation and experience. With the development of the closed laboratory activities and the emphasis on software engineering throughout our new curriculum, we believe we are providing students with the necessary software engineering education to build, enhance, and maintain quality software products for the multitude of modern computer applications.

The response and support of our students and the positive feedback we have received from campus recruiters tell us we are on the right track.