Introducing Software Engineering to the Freshman Student

Yi Liu, Wei Wang and Onyeka Ezenwoye
Department of Electrical Engineering and Computer Science
South Dakota State University
Brookings, SD 57007
{yi.liu, wei.wang, onyeka.ezenwoye}@sdstate.edu

Abstract

With most of the software engineering curriculum settings, the first Software Engineering course is not offered early enough to give the students a good overview of what software engineering is, in their early stage of study. It is not only unbeneficial to stimulate students' interest in software engineering, but also hinders the success of retaining the students in the program.

This paper describes a new freshmen level course – SE 102 Introduction to Software Engineering – that uses a hands-on setting to introduce freshman students to the beauty of software engineering. The authors design a semester long hands-on project using robotics to expose the students to the major software development phases, including requirements analysis, design, implementation, and testing. This paper outlines the approach and structure of the course, and presents the evaluation methods for assessing the course effectiveness.

This course is intended to boost students' motivation and interest in software engineering, and improve the program's recruitment and retention rates. Due to its generic design, this course can be seamlessly adapted to other software engineering programs in peer institutions.

1 Introduction

Since Imperial College started the first undergraduate Software Engineering program in 1985 [4], a number of universities in the world have established undergraduate software engineering degree programs. In the United States, there are over 50 software engineering undergraduate programs [8, 9].

Although the number of the programs has been increasing, especially in the past decade, software engineering undergraduate programs are still not as familiar to the prospective college students as computer science undergraduate programs are. This brings considerable challenges to recruitment. Challenges also exist in how to retain the students in the software engineering programs. With most of the software engineering curriculum settings, the first software engineering course is not offered early enough to give the students a good feel of what software engineering is in their early stage of study. It is not only unbeneficial to stimulate students' interest in software engineering, but also hinders the success of retaining the students in the program.

In order to address these issues, the authors propose a freshman level course *SE 102 Introduction to Software Engineering* that uses a hands-on approach that employs the use of robotics to expose the students to the major software development stages, including requirements analysis, design, implementation, and testing. The goal of introducing this course into the software engineering curriculum is to broaden colleague students' participation in software engineering program and further consider software engineering as their future career.

In this paper, the authors describe the approach and structure of the course, and the evaluation methods for assessing the course effectiveness. The paper is organized as follows: section 2 presents the overview of the undergraduate software engineering curriculum and the authors' experience of using robotics in an engineering class; section 3 specifies the objectives of the course and then illustrates the details of the course structure; section 4 presents the evaluation plan and section 5 concludes the estimated impact on the recruitment and retention.

2 Background

The undergraduate software engineering program was started in the Department of Electrical Engineering and Computer Science at South Dakota State University in 2003. In August 2010, the program was accredited by the Accreditation Board for Engineering and Technology (ABET) for the first time.

2.1 Overview of the Software Engineering Curriculum

The current software engineering curriculum requires 132 credits with 9 mandatory software engineering courses (total 26 credits), 8 mandatory computer science courses (total 24 credits), and 9 applied elective credits. The 9 mandatory software engineering courses include SE 305 Foundations of Software Engineering, SE 320 Software Requirements and Formal Specifications, SE 340 Software Architecture, SE 330 Human

Factors and User Interface, SE 420 Software Project Management, SE 410 Software Test and Quality Assurance, SE 440 Embedded Systems, SE 464 Senior Design I, and SE 465 Senior Design II.

The students do not take the first introductory software engineering course *SE 305 Foundations of Software Engineering*, which introduces the software engineering principles and major development process, until their sophomore year. However, the authors found out that quite a few students switch majors during their freshman year. These students lose interest in software engineering even before the first introductory software engineering course.

As the curriculum guideline [1] states, while certain software engineering topics that requires maturity should be taught towards the end of the curriculum, the introductory material on these topics can be taught early to facilitate maturity. From this perspective, the authors believe it would be beneficial to introduce the beauty of the software engineering to the students in their freshman year. Thus, the authors propose to add *SE 102 Introduction to Software Engineering* into the curriculum as a freshman level course.

2.2 Robotics in Education

Utilization of robotic technologies in students' science and engineering education has been widely accepted in the research and education community. For example, Parallax Scribbler robot [5] is used for computer science introductory courses by Institute for Personal Robots in Education (IPRE) at Georgia Tech University; LEGO Mindstorms robots [6] have been extensively applied to college freshman engineering experience [7].

CEENBoT [2], developed by the Department of Computer and Electronics Engineering at University of Nebraska-Lincoln, is a programmable platform to provide innovative robotics solutions and accessories to educators, hobbyists and researchers [3]. In Fall 2010, the Department of Electrical Engineering and Computer Science at South Dakota State University started using CEENBOT in the General Engineering course GE 101 to the Electrical Engineering, Computer Science and Software Engineering students. . The students were teamed up doing in-class exercises using CEENBOT robots and they gave positive feedback on the usage of CEENBOT robots in end-of-class surveys.

The authors believe CEENBOT robots are accessible to the freshman students and it would be a good approach to use it in the SE 102 class.

3 Course Design

SE 102 Introduction to Software Engineering is a one-credit course that will be offered in the spring semester of the freshman year. It is mandatory to the software engineering students and open to all other majors.

"Software engineering is about creating high-quality software in a systematic, controlled, and efficient manner. There is an important emphasis on analysis and evaluation, specification, design, and evolution of software. In addition, there are issues related to management and quality, to novelty and creativity, to standards, to individual skills, and to teamwork and professional practice that play a vital role in software engineering." [1]. To understand the details on the software development process, some familiarity that is often not available to the freshman student is needed. It is important for the student to learn what the major development phases are and how they are applicable in a simple real-world project. The *SE 102* is designed to cover four software development stages, including requirements analysis, design, implementation, and testing, to provide the students a big picture of what software engineering is and what topics they will learn in the rest of the studies. The students will further be introduced to software engineering principles in the *SE305* course in the sophomore year, and learn the details of requirements analysis, design, implementation, and testing in *SE 320*, *SE 340*, and *SE 410* in junior and senior years.

Following the curriculum guideline 10, which states that software engineering must be taught as a problem-solving discipline, and the curriculum guideline 17, which encourages using a variety of teaching and learning approaches beyond the lecture format [1], the authors design SE 102 to be a hands-on project based course that uses robotics for its technical settings.

3.1 Objectives and Outcomes

The goal of this course is to broaden colleague students' participation in software engineering program and further consider software engineering as their future career. The specific objectives of the course are:

Objective 1: expose the freshman to software engineering principles
Objective 2: stimulate the students to consider software engineering for continuing studies and careers.

The expected outcomes from this course are:

Outcome 1: The students have a big picture of what software engineering is.
Outcome 2: The students understand the major software development stages and are able to apply them into a real-world project using robotics.
Outcome 3: An 85% retention rate of first year students in software engineering program at SDSU is achieved.

Outcome 1 and 2 address the objective 1 and outcome 3 addresses the objective 3.

3.2 Course Structure

SE 102 is meets one hour every week with total 15 classes in a semester. It is a project-based class with CEENBOT robots used for the project.

3.2.1 The Project

As for the first time offering, a "robot traveler" project will be used through the course. The project requires each team to make the CEENBOT robot set out from Dr. Wang's office, visit Dr. Liu's office, drop by Dr. Ezenwoye's office, go to the computer lab, and then move back to Dr. Wang's office. All the places that a robot will visit are allocated in the same floor. The quality of each team's development is measured upon (1) the completeness of the project, i.e., whether all the requirements are satisfied, and (2) the quality of the travel, such as how long it spends to travel, and whether it hits wall/barriers on it way.

3.2.2 Course Topics

In SE 102, students will experience how to develop simple software in teams by applying the software engineering principles and following the software development processes. To make it accessible to the freshman students, the water fall model will be used to introduce the software development cycle and major development process. SE 102 covers the following topics:

- Analyze the requirements of a given project
- Design the architecture based on the requirements
- Implement the design
- Test the product

3.2.3 Course Delivery

A three-session set, one or two hour l*ectures*, one or two hour *labs* and *documentation*, is designed for delivering each topic.

Topic1: Analyze the requirements of a given project

The purpose of this topic is to introduce the students to the basics of the requirements stage and provide them a chance to experience the requirements elicitation and documentation.

The lecture will cover what the requirement stage is, why it is important, how to elicit the requirements and how to document requirements. Interview and requirements workshop will be introduced as elicitation techniques. As for the freshman student, formal Software Requirements Specification with IEEE format will not be introduced for documenting the requirements, but the students will be briefly introduced to use both informal and formal approach (such as use case diagrams and descriptions) to document the requirements.

A requirements workshop will be held in the lab session (one hour). During the workshop, the software engineering faculty members will be customers and the students will experience how to elicit the requirements by using the approach introduced in the lecture.

The deliverable for this topic is a requirements document for each team. In the document, the students will be required to illustrate the problem they are going to solve and specify the significant features that they collect from the workshop in nature language.

Topic2: Design the architecture based on the requirements

The purpose of this topic is to introduce the students to the basics of the design stage and provide them a chance to produce an architectural design.

A two-hour lecture will explain the students what design stage is for, and use examples to compare a good design vs. a bad design, and illustrate one architectural pattern – main program and subroutine. Also, the students will be introduced to draw a simple UML class diagram for documenting the design.

One hour lab will be held for discussion on how to design the "robot traveler" system according to the requirements.

Each team will deliver a design document using UML class diagram(s).

Topic3: Implement the design

The purpose of this topic is to teach the students how to implement system according the design.

A two-hour lecture will provide the syntax of programming with CEENBOT and use examples to show the students how to do the programming with CEENBOT.

Three hour labs will be held for the students to do programming.

The code from each team will be the deliverable for this topic.

Topic 4: Test the product

The purpose of this topic is to teach the students how to validate the system they build against the requirements.

The lecture will introduce what the software testing and software quality are, and one or two the basic approaches for making test cases.

Two hour labs will be held for this topic: one for the students to design the test cases and another for them to conduct the testing and debugging.

Each team will deliver the written test cases and the test report.

Dr. Liu will offer the lectures on requirements and design, Dr. Wang will offer the lecture on Implementation, and Dr. Ezenwoye will offer the lecture on Testing. All three faculty members will teach the labs.

The last class is for the students to present the systems they develop. A rubric (with 100 points) to measure the students' work is designed based on the two criteria that are specified in section 3.2.1 and each team's grade will be evaluated by using the rubric.

4 Evaluation Plan

SE 102 will be offered in spring 2012 for its first time. Three assessment tools will be used for evaluating the course effectiveness:

(1) An initial survey and a final survey will be used to assess *outcome 1* "The students have a big picture of what software engineering is".

A four-level likert scale will be used in the questionnaire. Three sets of questions will be asked in both initial survey and final survey. They are: how well they know the major development stages, how much the students are interested in software engineering, and whether they would consider software engineering as their major. Comparisons will be done to the results from both surveys. The criterion of successfully achieving the outcome is: 80% of the students answer they know the major development stages *well* or above (with the likert scale of "not at all", "a little", "well", and "very well"), and 80% of the students answer they are at leasts *interested* in software engineering (with the likert scale of "not at all", "a little", "interested", and "very interested").

(2) The results from the project rubric will be used to assess *outcome 2* "The students understand the major software development stages and are able to apply them into a real-world project using robotics".

The criterion of successfully achieving the outcome is: 100% of the students receive 60 points out of 100 points, and 70% of them receive 80 points out of 100 points.

(3) Census on the enrollment in software engineering will be made on the census day of fall 2012 for the *outcome 3* "An 85% retention rate of first year students in software engineering program at SDSU is achieved".

85% retention rate is the threshold for judging successfulness. If the retention rate is higher than 85%, the course successfully achieves the outcome. Otherwise, the course does not achieve the effectiveness as expected.

5 Conclusion

After reviewing the available curricula of other software engineering programs [8, 9], the authors found that almost all the programs offer their first introductory software engineering course in the sophomore year or even later, in the junior year. If other software engineering programs in peer institutions also have concerns on retention and recruitment, this course can be seamlessly adapted to them due to its generic design.

As for a broader impact, this course can be easily adapted to be a 3-day summer camp or workshop for high school students by removing some relatively advanced topics on

documenting and UML modeling. The authors will try this approach in high schools if the possible outreach is available.

The authors are looking forward to the success of this course after its first offer in spring 2012.

References

[1] ACM and IEEE Computer Society. *Software Engineering 2004Volumn*. http://sites.computer.org/ccse/, last accessed on March 16th, 2011.

[2] CEENBoT robot. http://www.ceenbotinc.com/ , last accessed on March 16th, 2011.

[3] CEENBoT robot. http://www.ceen.unomaha.edu/TekBots/, last accessed on March 16th, 2011.

[4] Finkelstein, A., European Computing Curricula: A Guide and Comparative Analysis, *Computer Journal*, vol. 36, no. 4, pp 299-319, 1993.

[5] Parallax robot. http://www.parallax.com, last accessed on March 16th, 2011

[6] Lego robot. http://shop.lego.com, last accessed on March 16th, 2011.

[7] S. Kim, J. Jeon, "Introduction for freshmen to embedded systems using LEGO Mindstorms," *IEEE Transactions on Education,* vol. 52, no. 1, pp. 99 - 108, Feb. 2009.

[8] Bachelor's in Software Engineering Degree Programs, Online and On-Campus. http://www.usnewsuniversitydirectory.com/bachelors/technology/software-engineering.aspx, last accessed on March 16th, 2011.

[9] Undergraduate Software Engineering Degree Programs. http://faculty.db.erau.edu/hilburn/se-educ/se-programs.html, last accessed on March 16th, 2011.