On Practice-Oriented Software Engineering Education

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

Damiano Distante
Research Centre On Software Technology
University of Sannio, Italy
distante@unisannio.it

Abstract

The old saying "practice makes perfect" has been proven to be true in many fields when training new people to master the skills needed for a particular domain. It is even truer for software engineering education. Software engineering, by and large, is the application of engineering to software. Unlike other disciplines in the computer sciences, only by coupling theory and practice can students (who usually lack real-world working experience) understand some of the abstract concepts and principles taught in software engineering courses. This paper summarizes our experience in teaching software engineering courses in two different universities using a practice-oriented approach that guides students through learning the different, and yet abstract, aspects of the software engineering process.

1. Introduction

Software engineering is an integral part of the computer science and engineering undergraduate curriculum. One main goal of software engineering curriculum is to teach students how to develop and manage large-scale, long-lived software systems in a cost-effective manner. Software engineering is the disciplined application of engineering principles to the creation of complex software systems. It is an amalgam of people, process, and technology [5]. Students need to learn both technical knowledge, such as requirements elicitation, architectural design, implementation, testing, etc., but also non-technical aspects, such as project management and development of know-how for a particular application domain. Software-intensive systems have become ubiquitous in every aspect of our living environment and have made tremendous impact in the global economy. Consequently, the way software engineering should be taught needs to change accordingly, to reflect the sheer demand for sophisticated software development. Traditional software engineering education that focuses heavily on software engineering methodologies is simply not adequate. In addition to continuing traditional computer science education [3], the changing landscape of pervasive computing requires new software engineering teaching focuses. For instance, how to develop safety-critical systems, secure software, and how to shorten transition time for students from classroom to industry, are examples of new foci.

When laying out the future of software engineering, Mary Shaw [4] pointed that the focus of the next decade for the education of software developers should be to "prepare students differently for different roles, infuse a stronger engineering attitude in curricula, help students stay current in the face of rapid change, and establish credential that accurately reflect ability." Towards these goals and to meet the demands of rapidly emerging software technologies, some new pedagogies of software engineering education have been introduced in the classroom, as well as new software development processes. For example, Laurie Williams [7] described the effects of using pair programming in an educational setting, Tilley et al. [6] has hosted a series of workshops that focus on software engineering course projects in different universities.

In this paper, we describe our experience and the lessons learned in teaching an undergraduate software engineering course at Florida Atlantic University in the USA and a

Proceedings of the 19th Conference on Software Engineering Education and Training Workshops (CSEETW'06)
0-7695-2647-0/06 $20.00 © 2006 IEEE

graduate Database System course at University of Lecce in Italy. The first course is called Principles of Software Engineering and is one of the core courses for both computer science and computer engineering majors. It is an introductory course that covers basic concepts and principles of software engineering includes both technical and non-technical aspects. The second course is one of the core courses in the M.Sc. in computer engineering at University of Lecce. The course is intended to provide students with knowledge on the techniques and technologies for designing relational databases and developing a software system to file data in it. The objectives of this course, and the approach used to teach them, makes this course mostly comparable to a software engineering course for this discussion.

In the following sections, our experiences in teaching these courses are described in three aspects: ownership motivation, professional tools practice, and fulfillment in solving real problem. Finally Section 5 summarizes the paper and outlines possible future work.

2. Ownership motivation

Software engineering and database systems course contents, such as process models, can be quite abstract for students who may lack real world working experience to understand them thoroughly, albeit some technical and concrete materials. In order to let students have a solid grasp of the course material, doing a course project is a good way to tightly couple theory and practice. Although there are both advantages and disadvantages in assigning students project topics vs. letting students pick their own subjects, from our experience in teaching the same courses several times having done both options, we have gained some good feedbacks in letting students pick their own project topics, whose experience which we called "ownership motivation". Some of the advantages of having students picking their own project topics (with the approval from instructors of the level of difficulty and substantial details) are discussed in the following.

2.1. Students leveraging their potentials and interests

When asked to pick their own projects, students can pick some topics in which they are interested, or they have wanted to do but could not find the proper environment to do them before, or they might pick something new that they want to learn. Since the students pick the project topics themselves, they have the motivation and desire to do a good job. By working in groups, "two heads are better than one", students are more confident in finishing the project since they share the working load as well as the risk of encountering unknown problems, and eventually find the solutions.

2.2. Students being their own architects

Since the students picked their own project topics, when it comes to present their project proposals, they already have some ideas of what they want to accomplish, such as what kind of user interface to use, and how users would interact with their final products. With the help of the instructor and a few iterative discussions, students are able to put their ideas into practice and realize their design by following what they learn from lecture material and using proper software tools. In other words, students are the architects of their own projects; the instructor is there to assist, rather than to manage, the projects. The benefit of being their own architects becomes clearer during different phases of the assignments. For example, when they work on their requirements phase assignment by using professional tools (see Section 3), it comes quite smoothly for students to define their own projects' functional and non-functional requirements, and define the traceability of these requirements. This experience is in stark contrast to our previous experience when students were given a project topic, where it was not as clear as what needs to be done unless they repeatedly asked questions from their customer (instructor), which could be an experience of its own.

3. Professional tools practice

One of the reasons Ariane 5 (a European rocket that carried commercial payloads into earth orbit) failed was that the system was not tested on real operational environment, but only in simulated environments [2]. The actual speed that was about five times faster than Ariane 4 was never tested in real environments, which caused a total loss of $500M. Although sometimes it is difficult to test a system in a real operational environment, we should do it as much as we can – admittedly, something not always possible for projects such as rocket launches.

Testing a system in a real environment is in a sense analogous to teaching software engineering in real world setting. In other words, we should give students the opportunities to experience real world practice whenever we can. The smaller the gap between software engineering education and real world practice, the easier students adapt to industry after they graduate. Not having students using software tools is not an option; having them using mock "toy" tools is not a good option either.

Following this philosophy, supported by IBM Corp. at Florida Atlantic University (FAU), we deployed a suite of professional software tools in our classroom assignments – Rational Suite Enterprise [1], a professional tool that includes virtually every aspect of software developments activities, from requirements to testing, to configuration management. Students get the first exposure to use this professional development tools in their assignments that are coupled tightly with the course material. This arrangement enables us to tightly couple theory and practice so that students have a very solid understanding of what they have learn in the classroom, and are able to put them into practice, i.e., their own projects. The feedback we received from students is very positive. Since most of the students who take this software engineering course are seniors, the deployment of professional tools in their learning experience is extremely beneficial to them. Because of their experience in this class, companies directly hired some of the students immediately after graduation.

4. Fulfillment of solving real problems

There is more incentive for students to finish high quality work when students know their project will be used in solving real problems and have impact on the society. The joy and fulfillment they have are something that could not be achieved if their projects were "toys".

South Florida has been subject to an unusually high number of tropical storms and fierce hurricanes for the past few years. Florida Atlantic University (FAU) has been using a paper system to manually file for Federal Emergency Management Agency (FEMA) damage claims. During the Fall 2005 semester, one of the groups in Software Engineering course did a project that helped FAU to digitize their damage claim forms and file FEMA claims electronically. By talking to the people at FAU and getting feedback of their requirements, students had a quite concrete goal in mind. They designed the user interface according to the actual damage claim form; they also automatically populated the claim forms of all FAU buildings geographic locations to expedite the data input process and reduce human input errors. In their final deliverables, the computerized filing form can also do a query on items that otherwise could not be easily done by filing manually, such as listing the top five most damaged items across the whole campus. At the end of the semester, when students gave a live demonstration, the fulfillment and pride the students had were beyond what their final grades could measure.

Similar experiences occurred at the University of Lecce in Italy where students developed their database system project for a real customer with a real schedule of production and a firm deadline to meet. Students of the database systems course were asked to form groups of two to four people and look for a real customer for their project or assigned one. Each group

of students was requested to develop its project in three main steps: 1) requirement elicitation, 2) design of the required database, 3) design and implementation of some functionality of the software system to access and use the database. The students were responsible for defining roles, tasks, and responsibilities inside each group, meet the project deadline (normally 45 days for each project), and report on the level of satisfaction of the customer when delivering the prototype they developed. Our experience in giving students the opportunity to develop a project, though not complete, meant to solve a real problem and answer a real need was extremely positive. Essentially all students given this possibility declared to be very satisfied and to think their experience will be useful for their future employment in industries. Most of them experienced for the first time a real working environment: a real customer, a real problem, and a real team.

5. Summary

As a professor, the most comforting moment is to see that students have learned something that is valuable to them and that they are making progress in their education. There is no doubt that there is more than one good way to teach software engineering. But what we have learned in our experience is to encourage students to act as the main players in the whole education process, rather than being told (or taught) all the time. To motivate students' interests and keep their enthusiasm going requires a new pedagogical methodology from the instructor side. From our experience, ownership motivation plays critical roles in students' learning experience. Deploying real-world professional software development tools in their assignments, which are tightly coupled with lecture material, is an effective way to shorten the distance between academic education and industry demands. Solving real world problems in students' projects can be the real incentive for students to finish high quality work and get a good sense of fulfillments by making a difference in society. Our teaching experiences from two different universities add another data point that practice is an effective way to shorten the gap between academia and industry.

Acknowledgements

Thanks for Ed Fernandez of Florida Atlantic University for his comments on early drafts of this paper, and to Scott Tilley of the Florida Institute of Technology for discussions related to software engineering education.

References

[1] IBM Rational Software, online at http://www-306.ibm.com/software/rational/.

[2] Lions, J., et al. "Ariane 5 Flight 501 Failure: Report by the Inquiry Board." European Space Agency, 1996.

[3] Parnas, D. "Education for Computing Professionals." *IEEE Computer*, 23(1), January 1999, pp. 17-22.

[4] Shaw, M. "Software Engineering Education: A Roadmap." *The Future of Software Engineering* pp 373-380, ACM Press 2000. Editor: Anthony Finkelstein

[5] Tilley, S. "Software Engineering 2." Online at www.cs.fit.edu/~stilley.

[6] Tilley, S.; Huang, S.; Wong, K.; and, Smith, S. "Report from the 2nd International Workshop on Software Engineering Course Projects" (SWECP 2005). To appear in *Proceedings of the 19th Conference on Software Engineering Education and Training* (CSEE&T 2006: April 19-21, 2006; Oahu, HI). Los Alamitos, CA: IEEE CS Press, 2006.

[7] Williams, L., and Kessler B. "The Effects of "Pair-Pressure" and "Pair-Learning" on Software Engineering Education." *Proceedings of the 13th Conference on Software Engineering Education & Training* (CSEE&T 2000, March 6-8, 2000). pp. 59-65.