Teaching Software Engineering Courses: When?

Nabil Arman and Khalid Daghameen
Palestine Polytechnic University
Hebron, Palestine
{narman,dkhalid}@ppu.edu

Abstract: Recently, we have noticed that there is an evident weakness among our students in applying and using software engineering principles in advanced courses that require major software projects. We have also noticed that many software graduation projects lack the concrete and correct usage of sound software engineering principles. In an attempt to determine the main reasons behind that, we reviewed the study plans of many IT-related departments, which generally distribute course over years and semesters, and found that software engineering courses are taught early in the plans. However, we argue that such courses should be taught as late as possible to emphasize the engineering principles rather than focusing on the details that are covered in other courses. We conducted a survey regarding that and the results were in favor of our argument.

Keywords: Software Engineering, Software Engineering Education.

1. Introduction

Many information technology (IT) majors/curriculums, including computer science, computer systems engineering, computer information systems to name just a few, include software engineering courses (at least one course) to teach students the software engineering principles. After reviewing the IT curriculums, we have noticed that software engineering courses are taught early in the study years, before a number of courses like database systems, operating systems, computer networks, etc.

Many issues regarding the teaching and practice of software engineering have been presented in the literature [1-10]. Some researchers focused on teaching of software engineering such as teaching in large groups or small groups. Odeh's argument was about teaching software engineering in large groups [7]. In our study, our argument is about the "time"/"when" of offering of a software engineering course. Some courses should be offered before a software engineering course and others must be offered after the software engineering courses.

In this paper, we argue that many supporting courses, like database systems, operating systems, networking essentials, to name just a few, to software development must be taught as early as possible and a software engineering course should be taught as late as possible to emphasize the sound engineering principles rather than focusing on the details that are covered in other courses. We conducted a survey regarding that and the results were in favor of our argument.

2. Motivation

As mentioned before, most students have weaknesses in applying the sound software engineering principles and best practices. In an attempt to find out the

main reasons behind that, we have decided to conduct this study to be able to specify why this is happening.

3. Teaching Software Engineering Courses

In teaching software engineering courses, many phases of software systems development are explained. For example, in system design phase many decisions have to be taken, including:

1. The decision regarding the system, whether it will be running as a centralized system or distributed. This affects the choice of the operating system to be used.

2. The decision regarding the number of tiers of the client/server system, whether it is one-tier, two-tier, or n-tier systems. This affects the whole implementation of the system. One tier will be the easiest to implement but the most constrained. Two-tier client/server system will be more difficult to implement, but it will have some well-known advantages. It will include a decision of the web-server to be used and the network protocol to be used. If the system is three-tier client/server system, decisions regarding the web-server, application server, and the database server have to be made.

3. The decision regarding the choice of the data model for the database, whether it is relational data model, object-oriented data model, object-relational data model, etc.

4. The decision regarding the choice of the programming language to be used in the implementation.

Generally, software engineering courses cover many details that are not supposed to be covered. For example, database modeling concepts are covered. If a database course was taken before the software engineering course, then there would be no need for such coverage. Instead, the focus will be on the general principles of the software engineering practices.

4. Analysis of the Questionnaire

Software Engineering course must be part of any computing program such as Computer Science, Information Systems, Computer Engineering, etc. This course must have Software Engineering concepts, theories, ideas, techniques that the student must have to endure his career.

Teaching Software Engineering course involve many things, from choosing the textbook, the material that should be taught, when it should be taught, and the instructor himself is part of the argument for software engineering course, which make it different from any other course.

To give an example for the textbook, let us take a look at one of the Deitel and Deitel books for teaching programming languages. Such textbooks contain comments and guides for software engineering. Teaching a programming language, using such textbooks for freshmen and sophomore students, is confusing compared to senior and junior students, as those are familiar with such comments.

If a programming language textbook discusses analysis and design for Object Oriented, it would make understanding the programming language itself harder and more complicated. So, the fact that the instructor should choose a book that not only matches the students' level but also concentrate on the subject matter.

Software Engineering material; system planning, system analysis, system design, system testing and maintenance, is the same for different methodologies: process oriented, data oriented or object oriented. In addition, there are a lot of other things that a software engineering course could have like formal systems, web systems, agile systems and prototype systems.

Many things of the above points deserve more attention. Lets take system plan which involves system size estimation, decision making, economic issues, tasks network, time plan, sources estimation, cost estimation, etc, this requires that students study some techniques in economics, like taking an economic engineering management course before software engineering course.

Another example, designing Object Oriented system is a lot different from Implementation of Object Oriented system using any computer programming language. Building Object-Oriented system comes in the late phases of Object-Oriented Engineering, which has requirements gathering, Objects Identifications, Object-Oriented design and Object-Oriented implementation (building). Therefore students should first learn what is Object-Oriented technology before starting to develop Object-Oriented systems.

As for the instructor, who could be a matter of debate, as most academics are rarely involved in software industry. An instructor who never developed a software system will have difficulty teaching students how to build software system. Teaching software engineering depends a lot on the concepts that have been implemented, so the instructor who has never been in software engineering industry would find it hard to teach software engineering as he/she lacks the experience and real world examples, as most of the examples in the book are still theoretical.

When teaching software engineering? This is a matter of debate and in this paper, we tried to answer this question. The rule of thumb for such is "What before how", for example you couldn't teach data structure before having the students a knowledge about a programming language. Which means that the curriculum must have a programming language course at least before getting into a data structure course. The same thing applies for software engineering course. There are many courses in the curriculum which should be taught before software engineering course and others should lag software engineering.

The questionnaire contains some of the questions about teaching a software engineering, such as:

1. Which course of the following have you used the concepts of the software engineering?
2. What is the percentage of the dependency of some courses on software engineering? and vice versa. Not to mention that the questionnaire contains also some introduction about the questioned person.

The questionnaire was conducted in Palestine Polytechnic University, the intended group of study was mostly senior and graduate students.

It is found that more than 95% of the students took the software engineering course in the third year, which means it is two years before graduating. The 5% of those had problems due to failure in some courses which cause the delay of taking that course. The curriculum in the PPU for Computer Systems Engineering has the software engineering in the third year.

An answer for the question that if they had taken the software engineering course more than once was "No", which means that they passed the course from the first time. We didn't intend to have such result, but it was like this. Also we found that most of the sample has grade good for software engineering course.

One of the questions that is mentioned above about which course did you use the software engineering concepts? The results were as follow:

Object Oriented: 5%
Graduating Project: 95%
Database concepts: 30%
Visual Programming: 26%
Algorithms design and analysis: 0%
Computer Graphics: 5%
Operating Systems: 5%
Digital Design: 0%
Systems Programming: 50%

The results were as we expected, the students use the concepts of Software engineering in the graduation project, which is offered in the last year. The issue with the 5% of the result was that those students were late and they didn't start their graduation project yet. As for both the system programming and visual programming, those courses include a simple project for the course and some students use the concepts of software engineering for such project. As for the other courses, it seems the students don't use the concepts of Software engineering which matches what we think of as to let the software engineering as late as possible.

Another question about getting the opinion of the students of when to teach a software engineering course, the results were shown in the table below:

Course name	Before SW	After SW	With SW
Data structure	95%	5%	
Object Oriented	65%	30%	5%
Graduating Project	0%	90%	10%
Visual Programming	50%	50%	
Computer Graphics	50%	50%	
Operating Systems	75%	25%	
Systems Programming	30%	70%	
Digital Systems	80%	10%	10%

Most of the computing curriculums should have some programming courses, operating systems, database concepts, networking essentials, variety of programming languages, computer basics and design. Each course has an impact on the students, for example the outcome of the programming language course is that the student must develop a program for a specific problem.

In our university, the students have the following courses in programming, like introduction to programming, in which student learn programming basics (programming in small), and some related courses such as data structure and algorithms design and analysis.

Programming is the implementation phase in software engineering life cycle, which should be known to the student before studying the life cycle of the

software development process itself, which matches fully the results of the questionnaire.

After the students were able to program in small, they should be able to program in large, which is supported by object-oriented course. The object-oriented programming course teaches the students the concepts of object-oriented technology using one of the programming languages like C++ or Java. The object-oriented course objective is equip them with the ability to analyze an existing object-oriented system, and able to modify such system. In this course, the student should first understand the object-oriented before they can apply and develop object-oriented solution. Later, if they are able to analyze a complete Object-Oriented system (which is software engineering), they can decide if the system is process oriented or object oriented. So an object-oriented course must precede a software engineering course, which supports our opinion.

It would be a good idea to have some programming courses at the end of the curriculum, those programming courses should be project-based courses, so that the students could implement their course projects in the new language and follow the steps of the software engineering, if the courses are not project-based, then it won't matter if such courses are leading or lagging software engineering. That what we see in the questionnaire when the students were in dispute about the visual programming course which is a programming course offered in the later years of the study. The dispute comes from the fact that the student weren't implementing the concepts of Software engineering as they aren't feeling the advantage of it.

Computer basics courses such as digital design and computer organization and architecture have no dependency on software engineering or vice versa. That was what we found in the questionnaire. Such courses would make no difference if the student learns it before or after software engineering.

System programming is one of the important courses in a computing discipline, or the course of such type like compilers design. Such courses should be project-based courses, so the students get benefit from a software engineering course taught before studying systems programming. The questionnaire results show that the system programming depends on the software engineering, and it should be offered with or after a software engineering course.

Database concept courses might be theory and implementation using any databases tools or computer language. The theory part includes the design of a database system and the implementation would be part of a complete system. From the study we argue that such a course should be offered at least with or before a software engineering course but not after.

The graduation project course, where the students should use all the concepts they have studied to develop a full-fledged software system/application, give the students the chance to use all what they grasped during the whole period of the curriculum. The students would apply software engineering principles step by step. The questionnaire also shows such fact.

5. Perceived Benefits

If students take most of the courses in an IT curriculum, a software engineering course will be taught with a concentration of the sound engineering principles. We advocate the idea that at least the courses Programming Languages, Object-Oriented Programming, Database Systems, Operating Systems and Networking should be offered before a software engineering course in a curriculum.

This means that a software engineering course should be offered as late as possible, just before a software graduation project (if the curriculum includes such a project). If the project is offered as an introduction to the project (in one semester) and a project (in the next semester) (i.e. over two semesters), the software engineering course should be offered with the introduction to the project to make the most out of the course, where students can apply all the principles they are learning to their graduation project.

6. Conclusions

The results of the questionnaire matches our argument regarding the time of teaching software engineering course in any IT-Related curriculum, our arguments is to teach software engineering course as late as possible. In our study we showed that with some arguments that some courses should precedes software engineering course and others should follow.

References

[1] Arman, N., and Manasra, G., "Software Engineering: An Established Engineering Discipline or Just a Computer Science/Information Technology Course," Engineering Education Conference, May 17-18, 2005, Palestine Polytechnic University, Hebron, Palestine.

[2] Conger, S., *The New Software Engineering,* 1st Edition, Course Technology Press, 1993.

[3] Pressman, R., "Software Engineering- A Practitioner's Approach", 5th edition, McGraw Hill , 2000

[4] Sommerville I., *Software Engineering,* 6th ed, Addison-Wesley, 2001.

[5] McCauly, R. and Lackson, U., "Teaching Software Engineering Early-Experiences and Results,"

[6] Mittermeir, R., "Teaching Software Engineering in the large universities",

[7] Odeh, M., "A Reflective Approach to Improve Learning and Teaching of Software Engineering in Large Groups", The International Arab Journal of Information Technology, Vol. 1, No. 0, July 2003

[8] Newble, D., and Cannon R., *A Handbook for Teachers in Universities and Colleges,* Kogan Page, 1991.

[9] Favela J. and Pena-Mora F., "An Experience in Collaborative Software Engineering Education," *IEEE Software,* pp. 47-52, March/April 2001.

[10] Bennett, S., McRobb, S., and Farmer, R., "Object-Oriented Systems Analysis and Design using UML", 3rd ed, McGraw Hill , 2006