Chapter 94
Teaching Basic Software Engineering to Senior High School Students

Barbara Köhler
Technische Universität München, Germany

Michaela Gluchow
Technische Universität München, Germany

Bernd Brügge
Technische Universität München, Germany

ABSTRACT

Software Engineering (SE) is an increasingly important topic as software projects increase in size, budget, and duration. The authors suggest starting teaching SE already to high school students instead of waiting until their freshman year at university. This chapter shows the principles the authors used for creating such courses. First, the authors explain which software lifecycle model the authors use, why, and how it needs to be tailored for students with very little development experience. Second, the authors discuss the educational models the authors apply to increase motivation and counter the inert knowledge problem often observed in lectures. The authors mainly focus on goal-based scenarios and scaffolding, two constructivist design methods. Finally, the authors present a case study of one course they conducted in fall 2011 with eleven high school students between ages 16 and 18.

INTRODUCTION

Software Engineering (SE) is a topic that has grown more important over the years as software projects increase in size, budget, and duration. This is why it takes up an increasingly large part of studies in Computer Science. But it does not only take up more time, education in Software Engineering now also starts earlier.

Teaching Software Engineering is hard, not only because of the breadth and complexity of the subject, but also because of the following inherent Catch-22 teaching problem: On the one hand, for students to really be able to understand and reflect on SE theory, they need to have some experience of working in a "real" project. On the

DOI: 10.4018/978-1-4666-4502-8.ch094

other hand, to learn successfully from a SE project course it is vital to have a sound understanding of the basic principles and practices of SE theory (Broman, 2010).

This is why we think that teaching software engineering on a small, personally meaningful student project, teaching the knowledge required for the next steps along the way and reflecting on it at the end of the project is a good way to teach software engineering. In past courses we have found that working on a project with the goal of developing a working product not only leads to a level of motivation we have not seen before, but also that students deepened their understanding of theoretical concepts. Incrementally building on this newly acquired understanding, new software engineering concepts can be taught.

In this chapter, we describe the creation of such a course from several perspectives. First we analyze which software lifecycle model can be taught in a course for students completely new to software engineering. We also explain how we help students to distinguish different software development phases and which phases might need to be taken over by the instructor.

In addition to the software engineering content to be taught we talk about the educational models used in our courses. Those models are all applied with the goal of increasing motivation while at the same time providing a more active and efficient learning environment.

After the theoretic basis we show the success of our course design in a short case study. In this study we prove that it is possible to develop a complete software project with high school students between ages 16 and 18, who do not have any prior programming or software engineering experience.

At the end of the chapter we make suggestions how to adapt our design to a traditional school classroom setting instead of a short course of a few days at a university. We discuss differences between the two settings and how to overcome difficulties of such a project course within school.

COURSE DESIGN THEORY

In our course we pursue two main objectives: First we want students to gain transferable knowledge about software engineering and basic programming skills. Second we want them to be motivated, eager to learn more about software engineering. Every design decision should be made in accordance with those goals while also taking into account our target group: senior high school students or freshmen, who have little or no knowledge about programming and software engineering. In the following section we discuss the major theories applied in designing the course. This includes theory behind principles from educational technologies as well as Software Engineering.

Motivational Psychology

When it comes to learning, motivation plays a big role. It has been shown that motivation is a great indicator of performance. In the long run a motivated student can outperform a more talented student that is less motivated. This is simply due to the fact that a more motivated student is more willing to spend time on a topic. This practice then leads to better performance. As evidence suggests that motivation is a crucial part in the learning process, the question now is: how can we motivate people?

For this we want to have a look at what is required to motivate people. In Figure 1 we see that motivation has two components. A person should be inherently interested in a topic or problem. “If a person lacks volition she will feel lost, bored, or disconnected from the task at hand. They can’t see why an activity or behavior is worthwhile” (Dignan, 2011). However while the want to actually do something is an important prerequisite it is not enough to keep a person motivated. Additionally the belief that we have the skills and tools to solve the problem at hand is required. When a person constantly feels overwhelmed by a situation, thinking it is too hard to deal with or if it is just