Chapter 2
Learning Software Industry Practices With Open Source and Free Software Tools

Jagadeesh Nandigam
Grand Valley State University, USA

Venkat N. Gudivada
Marshall University, USA

ABSTRACT

This chapter describes a pragmatic approach to using open source and free software tools as valuable resources to affect learning of software industry practices using iterative and incremental development methods. The authors discuss how the above resources are used in teaching undergraduate Software Engineering (SE) courses. More specifically, they illustrate iterative and incremental development, documenting software requirements, version control and source code management, coding standards compliance, design visualization, software testing, software metrics, release deliverables, software engineering ethics, and professional practices. The authors also present how they positioned the activities of this course to qualify it for writing intensive designation. End of semester course evaluations and anecdotal evidence indicate that the proposed approach is effective in educating students in software industry practices.

INTRODUCTION

Software Engineering (SE) courses are some of the most challenging ones to teach in Computer Science (CS) curricula. Not only do students need to learn basic concepts, principles, and methods, but also master industry practices and tools in these courses. Lecture-based approaches to espousing software engineering principles hardly engage students' attention (Nandigam, Gudivada, & Hamou-Lhadj, 2008). Students often view software engineering principles as mere academic concepts and graduate without a clear understanding of how these principles can be used in practice. By the time students take their first SE course, it is quite unlikely that they have written programs that are more than 500 lines long.

DOI: 10.4018/978-1-5225-3923-0.ch002

It is also equally unlikely they had an opportunity to inspect large programs (> a few thousand lines of code) written by others.

One practice that seems to pervade across universities to bringing software engineering professional practices into the classroom is using a semester-long term project. In this project, students are expected to demonstrate their ability to apply software engineering practices and tools in solving a real-world problem in a team environment. However, there is no established approach to accomplishing the above goal due to various factors discussed below.

Selecting a right project with appropriate scope is in itself a challenge. In our experience with teaching SE courses, asking student teams to self-select a project rarely produces successful outcomes. Students typically overestimate or underestimate project scope and complexity. Overestimation leads to selecting a trivial project and embellishing it with superficial complexity. Underestimation results in switching to a trivial project halfway through the semester. In either case, the project scope and complexity are insufficient for students to fully experience professional software development practices.

The overarching goal of this chapter is to present our approach to teaching software engineering industry practices and tools in the backdrop of SE concepts, theory, methods, and principles. Using suitable software tools and team projects, we promote conceptual understanding and practical skills of the following topics: role of tools in the software development life cycle; iterative and incremental development as a means for timely project completion; requirements elicitation and specification; source code management with version control; importance of adhering to coding standards; design visualization; verification and validation through software testing; measuring and using software metrics as a means for improving software quality; software release management; ethics and professional practice; and writing as a means to learning. We also discuss how Open Source code bases can be used in achieving the above learning goals.

SOFTWARE ENGINEERING COURSE

Our undergraduate SE course includes a semester-long (about 14 weeks) software development project to provide students hands-on experience with processes, methods, techniques, and tools of software development. The course first provides the necessary theoretical foundation for a broad range of topics – software engineering process models, project management, software requirements elicitation and specification, use case modeling, UML, object-oriented analysis and design, design patterns, test-driven development, version control, system building, software testing, mock object frameworks, software maintenance, software internationalization, SE ethics, and writing skills. Though the topics are quite a few, very focused and conceptually oriented lectures make this task possible. Students gain practical aspects of these topics by working on a realistic project in a team environment.

Students begin the course by writing a short formal paper on a SE ethics topic. The semester-long project involves development of a software product using an iterative and incremental development model. Students use Eclipse IDE (n.d.), and several free and open source tools and plugins available for the Eclipse IDE. The product is delivered incrementally in three releases with each release taking roughly 4 weeks of effort. The course also includes a midterm, a final exam, and several quizzes as part of formative and summative assessments. The weight distribution of various components in the course is: term paper (10%), ethics writing assignment (5%), term project (30%), midterm exam (20%), final exam (25%), and quizzes (10%).