Chapter VIII
Teaching Software Engineering in a Computer Science Program Using the Affinity Research Group Philosophy

Steve Roach
The University of Texas at El Paso, USA

Ann Q. Gates
The University of Texas at El Paso, USA

ABSTRACT

This chapter describes a two-semester software engineering course that is taught in a computer science program at the University of Texas at El Paso. The course is distinguished from other courses in that it is based on the Affinity Research Group (ARG) philosophy that focuses on the deliberate development of students' team, professional and technical skills within a cooperative environment. To address the challenge of having to teach professional and team skills as well as software engineering principles, approaches, techniques, and tools in a capstone course, the authors have defined an approach that uses a continuum of instruction, practice, and application with constructive feedback loops. The authors hope that the readers will benefit from the description of the approach and how ARG components are incorporated into the course.

INTRODUCTION

The Computing Curricula 2001 (CC2001) project is the product of a joint effort by the Computer Society of the Institute for Electrical and Electronic Engineers (IEEE-CS) and the Association for Computing Machinery (ACM) with the goal of developing curricular guidelines for undergraduate programs in computing. CC2001 describes a set of recommendations for undergraduate programs in computer science (CS) and has had significant influences on curriculum development throughout the world (ACM, 2004). It includes the following statement with respect to the project

component of a Computer Science Curriculum (CC2001, 2001, p. 45):

The course descriptions . . . offer several models for including project work in the curriculum. The first strategy is simply to include a project component as part of the required intermediate or advanced course that covers the core material on software engineering. This strategy is illustrated by the course CS292{C,W}. Software Development and Professional Practice, which includes a team project along with a significant amount of additional material. As long as students have sufficient time to undertake the design and implementation of a significant project, this approach is workable. The projects in such courses, however, tend to be relatively small in scale, simply because the time taken up by the software engineering material cuts into the time available for the project.

All accredited software engineering programs and almost all accredited CS programs in the United States have a capstone experience in the undergraduate curriculum (CC2001, 2001; EAC, 2007; CAC, 2007). Like many other CS programs, the CS program at the University of Texas at El Paso (UTEP) combines the project experience with an introduction to software engineering principles. The two-semester sequence is taken in the students' final year of study and focuses on fundamental software engineering topics while developing the students' communication and team skills, establishing a venue in which to engage in meaningful discussions about the Software Engineering Code of Ethics and Professional Practice (ACM/IEEE-CS, 1999), providing practical experience, and supporting faculty-student interaction.

Teaching a capstone course in a software engineering program, where students have had significant exposure to software engineering concepts prior to entering the course, and teaching a capstone in a CS program, where students have usually had no prior software engineering courses, are manifestly different from each other. As noted in the CC2001 report, teaching the software engineering material and having students work together in a project setting is challenging. UTEP has met this challenge by developing a course that focuses on the practice of software engineering in a project that involves actual clients and the *deliberate* development of professional skills as espoused by the Affinity Research Group (ARG) model.

The primary goals of the UTEP course are to provide students with (a) a fundamental and functional understanding of the methods, tools, and techniques required of rigorous software engineering so that they can identify and adopt the practices needed in the workforce; (b) the experience of working with an actual client to develop a product so that they can learn to manage issues, such as incomplete, ambiguous, changing and inconsistent requirements, and to deal with time pressures; (c) the ability to apply software engineering principles to a software project; (d) the ability to prepare documentation in adherence to IEEE standards; and (e) the experience of working effectively in teams.

The UTEP approach is unique in that it uses the Affinity Research Group (ARG) model (Gates, 1999; Teller, 2001; Gates, 2007). The two principal tenants of the ARG model that apply to software development teams in the academic setting are the cooperative learning paradigm and the structured, intentional, and deliberate development of professional and technical skills. The ARG model has processes for evaluating work products and iteratively revising them. These processes have been adapted for use in the capstone project course.

In this chapter, we describe the techniques and approaches to teaching software engineering that we have developed and used for the past decade. Our philosophy, derived from the ARG model, is to focus on the development of each student.