Chapter 6.7
Teaching Globally Distributed Software Development (DSD): A Distributed Team Model

Stuart Faulk
University of Oregon, USA

Michal Young
University of Oregon, USA

ABSTRACT

There is a growing demand for computer science graduates with the skills necessary to work effectively in the global context. Experience suggests that the best way to learn the necessary socio-technical skills is though participation in collaborative project courses where the student teams are actually globally distributed. However, this pedagogical model has proven difficult to disseminate or sustain due to high adoption costs and the difficulty in finding teaching partners.

This chapter describes an approach to building a collaborative teaching community that seeks to address these problems. It begins by identifying the skills students should acquire in a Distributed Software Development (DSD) course and discusses why firsthand experience with DSD problems is essential to learning them. The chapter identifies the attributes that make DSD project courses difficult to develop or teach, and then it describes a distributed team approach to developing a reusable infrastructure and a teaching community to address those difficulties. Future work focuses on building an international community of educators and industry participants interested in partnering to develop and teach DSD courses.

DOI: 10.4018/978-1-61350-456-7.ch6.7

INTRODUCTION

As the software industry increasingly deploys globally distributed teams for software development, there is commensurate demand for computer science graduates with the skills necessary to work effectively in the global context. This includes not only the software engineering skills needed to manage the technical problems of Distributed Software Development (DSD), it also includes the human skills required to address the issues of managing or working in project teams that span languages, cultures, and time zones.

While software development has become a global enterprise, computer science education remains largely parochial in course content and student experience. Most students are introduced to software development, project management, and teamwork in software engineering project courses where the teams are formed from their classmates and development proceeds by face-to-face interactions among peers. In contrast, industrial experience shows that "distance matters" in the sense that geographic, temporal, and cultural differences result in software engineering coordination and control problems that differ both qualitatively and quantitatively from those of co-located developments. Where students work in co-located teams, many of these problems simply do not arise. Moreover, informal communication tends to fill the gaps left by problems like unclear requirements, ambiguous specification, or poor planning. As a result, few students have the opportunity to encounter the kinds of problems that arise in DSD or apply their skills to development issues beyond programming.

A pedagogical model that promises to provide a more globally relevant experience is one in which student teams are actually geographically distributed. A few universities have implemented distributed software engineering project courses with domestic partners or universities in other countries. Under this model, development teams are comprised of students from different schools who must collaborate to complete a common software project. Results of these efforts suggest that, even for student projects, the need to collaborate at a distance introduces the same kinds of communication and control issues endemic in real distributed developments. Further, the lack of informal communication channels requires students to apply more formal methods in their specification, design, and project planning.

While effective, this pedagogical model has proven difficult to disseminate or sustain. The adoption barrier is high relative to traditional project courses. In addition to the usual pedagogical materials, distributed development requires significant infrastructure to support student collaboration on projects, as well as faculty collaboration on teaching and managing the course. There are additional logistical issues in coordinating term schedules and institutional administration. Further, the collaborative teaching model requires reliable and available teaching partners to ensure the course can be institutionalized in the curriculum. Currently, this depends on the whims of individual instructors. The net result is that industry's need for students educated in distributed software development is rapidly outpacing what universities can provide.

We are currently engaged in a three-year, NSF-funded effort focused on lowering barriers to adoption and improving student access to project courses teaching DSD skills. Our work suggests that the difficulties of developing and teaching collaborative DSD project courses can be overcome using a community-based, distributed team model. In this model, faculty from a community of geographically distributed institutions work together to develop a common set of pedagogical materials, then partner in twos or threes to teach concurrent DSD project courses. The goal, over time, is to create a growing community of university, faculty, and industry participants who contribute to the common set of pedagogical materials, develop infrastructure, and provide