

A SOFTWARE ENGINEERING PROJECT COURSE MODEL BASED ON STUDIO PRESENTATIONS

Thomas B. Horton¹ and John C. Knight²

Abstract - We present a model for a software engineering project course that has proven successful as the capstone of our required sequence of software courses in our computer science major. It differs from many similar courses because all teams work on the same project, defined by the instructor (not projects for "real" customers). The course's project centers on weekly "studio presentations" during which student teams present preliminary results and receive immediate feedback. This model allows us to efficiently offer a project course for over a hundred students. The nature of the project itself supports other course goals. We believe that our project course model could be adopted and used effectively in computing and other disciplines.

Index Terms – laboratories, project courses, software engineering education, studio presentations.

Software engineering is often taught to undergraduates in part by means of a project-oriented course, where student teams create some number of life-cycle deliverables. Such courses challenge instructors (as well as students), especially in terms of evaluating student work products and communicating how they might improve their work.

A student should gain skills and insights that are typically acquired through experience. Indeed, many aspects of our discipline require expert judgment that can be learned best only by interaction between a student and teacher that is of the *master/apprentice* form. In design, for example, the difference between a good software design and a bad one is often subtle and very application specific. Only after having his or her mistakes with design pointed out by an expert does the student really gain the right type of judgment. The master/apprentice approach does not scale well where there is only one master but dozens of apprentices as is typically the case in a modern university.

We have developed an approach to teaching the "experience" element of software engineering that centers on a teaching paradigm that we call *studio laboratories*. This attempts to provide as much of the master/apprentice relationship as possible in a context where the apprentices vastly outnumber the masters. The studio laboratories are

supplemented by a number of other techniques that, when taken together, create a novel teaching environment.

The model for this course has some interesting and significant differences from many other software engineering project courses. We do not attempt to find a "real" customer or simulate such a stakeholder. Student teams work on the same project, and the project only varies slightly from year to year. But this allows us to derive a number of benefits from the following course components:

- Weekly laboratory meetings are comprised of *studio presentations*: student-teams present preliminary results for a deliverable, which is immediately critiqued by the instructor and other students. Presentations by five or six groups usually reveal all the major problems and allow several candidate solutions to be reviewed. Teams use what they learn to refine their work into a final solution due the following week.
- The studio presentation approach together with the single-project approach allows this course to scale to large classes (e.g. 130 students).
- The software project involves specialized hardware items (e.g. a control system for a mobile robot, or a system to manage users accessing a remote Web-camera). The hardware component enhances the project experience greatly because it introduces diverse challenges (constraints, real-time performance, etc.) and adds immense interest.
- Small teams are paired to build a client-server system, each developing half of the system. This requires them to overcome team and system integration issues.
- Students are motivated by the prospect of demonstrating their working systems to their peers, and in the case of the robot systems, by competing in a contest of "robot games".

The model has been used for a similar software engineering project course at another university. Positive feedback on both versions of the course indicates that this model can be effective when adopted and modified for other situations. Current work in progress seeks to evaluate more rigorously whether certain student performance goals are met, as well as to introduce studio laboratories in other courses in our program.

¹ Thomas B. Horton, Department of Computer Science, University of Virginia, horton@cs.virginia.edu

² John C. Knight, Department of Computer Science, University of Virginia, knight@cs.virginia.edu