The Software Studio in Software Engineering Education

Sarah Kuhn
Dept of Regional Economic and Social Development
U. Massachusetts Lowell
Lowell, MA, USA 01854
Sarah_Kuhn@UML.edu

Orit Hazzan
Department of Education in Technology and Science
Technion – Israel Inst. of Technology
Haifa, Israel 32000
oritha@techunix.technion.ac.il

James E. Tomayko
School of Computer Science and the Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA, USA 15213-3890
jet@cs.cmu.edu

Bruce Corson
Corson Associates /Architects
Sebastopol, CA, USA 95472
bacorson@sonic.net

Abstract

Four panellists, representing a variety of backgrounds and perspectives, discuss the use of "studio" teaching in software engineering and computer science education. Taking its inspiration primarily from the studio tradition in the training of architects, the software studio offers students an intensive, project based, multifaceted experience that integrates many competencies. The panellists explore various aspects of and experiences with studio education, with particular emphasis on Donald Schön's concept of "reflective practice."

Introduction

The studio is the central training method in architecture schools and it is considered to be a suitable educational environment for design studies in general. The architecture studio has been central to architectural training for most of the twentieth century. "Studios are typically organized around manageable projects of design, individually or collectively undertaken, more or less closely patterned on projects drawn from actual practice. They have evolved their own rituals, such as master demonstration, design review, desk crits, and design juries, all attached to a core process of learning by doing." ([4], p. 43).

In such studios students develop projects with the close guidance of a tutor. Usually, a group of students meet their tutor in the studio several times each week. Such an intensive schedule puts pressure on the students and, as a result, they devote themselves to the art of design: students learn the skills, the professional language and the discipline's ways of thinking. In addition, this learning environment exposes students to many kinds of social interactions, such as: work on team projects, contribution to class discussions, and presentation of their design-products in front of their classmates.

An analysis of the kind of tasks that architecture students are working on, and a comparison of these tasks to the problems that software engineering students are facing, suggests that the studio may be an appropriate teaching method in software engineering departments as well. Indeed, several attempts to adopt the studio for software engineering education have been conducted (Cf. [1], [2], [5], [6], [7], [8]).

One common idea behind most of the software studios is the idea of ‘reflective practice’, introduced by Donald Schön ([3], [4]), that guides professional people (architects, managers, musicians and others) to rethink and examine their professional creations during and after the accomplishment of the creation process. The working assumption is that such reflection improves proficiency and performance within the professions. However, there are various ways and approaches by which a software studio can be implemented.

The panellists describe their perspective on the ‘software studio’ based on their own experience. Our goal in the panel is to encourage conversation among people who are exploring studio teaching in CS—including those teaching project courses who may want to move in the direction of studio teaching—with an eye to advancing pedagogy and scholarship in this area. As part of the discussion, we hope also to consider possible future actions, such as forming a working group or other means for continuing and supporting the work about software studios.

Sarah Kuhn

Characteristics of the architecture studio include: project-based work on complex and open ended problems, very rapid iteration of design solutions, frequent formal and informal critique, consideration of a heterogeneous range of issues, the use of precedent and thinking about the whole, the creative use of constraints, and the central importance of design media. An explicit attempt at the Massachusetts Institute of Technology to adapt the architecture studio to the teaching of software design reveals much about the similarities and differences of the two professions. I will discuss MIT’s experimental course, and other related attempts, and their implications for CS education.

Orit Hazzan

In my presentation I explore similarities, as well as differences, between the architecture studio and the software studio. The analysis is based on analysis of the reflective practitioner perspective and of the profession of software engineering, visits to architecture studios, and conversations with tutors in architecture studios and with computing science practitioners. Specifically, I examine adjustments and modifications needed to be done, in order to adopt the studio from the architecture schools into software engineering programs.

James E. Tomayko

Carnegie Mellon University's Master of Software Engineering professional program has had the Software Development Studio as the centerpiece of the curriculum for 12 years. The Studio was explicitly based on architectural studios and the work of Donald Schön, emphasizing reflective practice. Over 200 experienced students (average 5 years) have gone through the program. The Studio features a group project with a real client lasting a year. Clients have included NASA, Charles Schwab, PPG, Boeing, Westinghouse, Carnegie Works, E-transport, the U.S. Air Force, and the SEI, among others, some clients sponsoring several Studios. Each is a proof-of-concept, or prototype, thus off the critical path. The presentation will be about how we achieve 'reflective practice' in order to teach our students.

Bruce Corson

Traditionally, we view “design” in hindsight—from the one-dimensional perspective of the (designed) artifact—the “delivered system”. There are, however, two other critical dimensions of design—the “delivery system” and the “discovery system”. “Reflective practice” is emerging as a potent means for approaching the “delivery system”—the environment in which we generate, assess and deliver design options. But it is in the third dimension of design—the "discovery system"—that we find the foundation of truly elegant, robust design. Here we seek to maximize the diversity of perspectives we bring to observing the "messy, indeterminate situations" from which "the problem"—the focus of the "delivery system" and "delivered system"—will be named and framed. Here we seek to understand the purpose of our design work—what we care about and why, who shares our reality and who has a different reality. In this realm we devote ourselves to fostering the most fertile environment possible for bringing forth every individual's ideas and perspectives. Ironically, it is this dimension of design that is most impeded by the very nature of academia and most organizations. Observations from academic and organizational explorations offer guidance for strengthening the "discovery system" and effectively pursuing the (dimensionally unified) "design of design".

References

[1] Docherty, M., Sutton, P., Brereton, M. and Kaplan, S. An innovative design and studio-based CS degree, *The Proceedings of the Thirty Second SIGCSE Technical Symposium on Computer Science Education*, Charlotte, NC, (2001), 233-237.

[2] Kuhn, S. (1998). The software design studio: An exploration. *IEEE Software* 15(2), pp. 65-71. http://www.computer.org/software/so1998/s2065abs.htm.

[3] Schön, D. A. (1983). *The Reflective Practitioner*, BasicBooks,

[4] Schön, D. A. (1987). *Educating the Reflective Practitioner: Towards a New Design for Teaching and Learning in The Profession*, San Francisco: Jossey-Bass.

[5] Tomayko, J. E. (1991). Teaching software development in a studio environment, *SIGCSE Technical Symposium*.

[6] Tomayko, J. E. (1996). Carnegie-Mellon's Software Development Studio: A five-year retrospective", *SEI Conference on Software Engineering Education*, http://www.contrib.andrew.cmu.edu/usr/emile/studio/coach.htm.

[7] Wallingford, E. *Software Studio*, (1998), http://www.cs.uni.edu/~wallingf/teaching/162/studio.html.

[8] Winograd, T. and Tabor, P. (1996). Software Design and Architecture. In Winograd, T. (ed). *Bringing Design to Software*. New York: ACM Press.