

Teaching Statement

Martin J. Schedlbauer

Teaching is not only a means for transferring knowledge but also a path for personal intellectual growth. I started teaching in 1988 as a Teaching Assistant while in graduate school and shortly thereafter as a corporate trainer for Data General. I was immediately hooked by the excitement of the intellectual stimulation of the classroom and the immediate impact a teacher has upon his students. This is particularly true for corporate training, where the transfer of practical knowledge is paramount. Ever since my first seminar during the summer of 1990, I have been authoring and delivering corporate training seminars on numerous software engineering subjects. I enjoy every minute of working with students and have started to teach academic courses as an adjunct instructor upon completing my doctorate.

TEACHING EXPERIENCE

My teaching career started in 1988 while working as a teaching assistant during graduate school. I occasionally lectured in an undergraduate computer graphics course and lead a summer graduate seminar on visual languages. Concurrently, I started teaching X/Motif and C++ programming seminars to professional programmers. Once I completed my graduate degree in 1991, I authored additional corporate training courses on object-oriented analysis & design, various programming languages, distributed software architecture, web development, client/server architectures, business analysis, requirements planning, software project management, and database design. I founded several training companies and taught two to four 40-hour courses each month on a variety of software development topics for about 17 years. I have attended many courses and conferences on training and education and have worked with a professional speaking coach during my time as a corporate executive. My teaching style is energetic, engaging, and very well received by students as evidenced by hundreds of very positive evaluations.

TEACHING PHILOSOPHY

Many corporate training practices also apply to academic teaching, such as creating a dynamic and collaborative learning environment in which the transfer of knowledge from the instructor to the students and from student to student thrives. I learned that a successful instructor must clearly explain the relevancy of the material being studied, give lucid presentations, excite the students about the process of discovery, and teach students how to learn rather than simply accumulate facts that may not be relevant in a short time. The latter issue is particularly important in technical fields where the rate of innovation is so rapid that facts studied during the freshman year may not be applicable upon graduation. As educators, we have to teach students how to learn and use their basic understanding of the principles to quickly adapt to new technologies.

I have noticed that students appear to be less engaged in computing than they were even ten years ago. For instance, fewer students stay up late to write new programs, build systems, and generally explore. Perhaps this is due to the inaccessibility and complexity of the new

operating systems and computers. I believe that we need to encourage students to explore and discover. One approach is the use of contests and collaborative projects. I have studied and practiced agile techniques and feel that many of the principles of collaboration, pair programming, and rapid prototyping can be successfully used in an academic environment.

As a corporate executive of several consulting organizations, I have seen first-hand the shortage of talented software developers and computing scientists. I am also keenly aware of the reduction in enrollments in information technology programs. Fewer students are interested in majoring in computer science for fear of not getting a job. The much hyped offshoring movement has compounded the issue. Companies are moving jobs offshore for lack of local talent; yet offshoring has caused students not to go into computing exacerbating the shortage for years to come and creating a proverbial “catch-22”. It is incumbent upon educators in the technical fields to make knowledge broadly accessible and to offer computing and technology courses to non-majors. It is also critical that we do what we can to assure that as many students as possible complete their degrees.

Often, the lack of good courseware and learning materials can affect how well students learn and whether they can complete a course. I have developed over 50 courses for corporate education where well-designed, high-quality courseware is essential for success. I believe that we can apply many of the courseware design patterns and heuristics to undergraduate and graduate courses as well. These best practices include good presentation materials, well-thought out examples, frequent learning checkpoints, appropriate assignments that reinforce and deepen the material, and well managed agendas. Students need to know what is expected of them and how they can do well. The instructor needs to be accessible and guide the student through the process of learning new material.

TEACHING QUALIFICATIONS

Given my strong presentation and classroom management skills as well as my industry experiences and academic training, I believe I would be qualified to teach any core computer science course at both the graduate and undergraduate level. Based on my corporate teaching background and my research interests in Human-Computer Interaction and Software Engineering Methodologies, I believe I am particularly qualified to teach courses in these areas:

- Human-Computer Interaction
- Software Engineering
- Large-Scale Software Architectures
- Object-Oriented Analysis & Design
- Graphical User Interface Development
- Software Project Management
- Internet and Web Systems
- Database Design and Architecture
- Introductory Computer Science Courses, including courses on algorithms, data structures, and programming in C/C++, Java, Visual Basic, and C#.

TEACHING STYLE

My teaching style has been refined over the years. I have taught courses throughout the U.S. as well as in several European and Asian countries. The levels of my audiences have ranged from recent college graduates to seasoned software developers. I have found that to be successful, an instructor must be well prepared, engage the students, foster a collaborative and active learning environment, establish fair and clear grading policies, set realistic goals, and identify and address misunderstandings early. All students generally want to do well and an instructor must provide an environment in which all students can succeed. Grading must be fair and be established at the beginning of a course. However, grading should not stifle the creativity of students and unconventional approaches should be encouraged. Exploration and discovery are essential to learning. We need to let students make mistakes and teach them how to learn from them.