Evidence of Teaching Effectiveness

Introduction

The effectiveness of my teaching is based on my classroom experience as an adjunct instructor at St. Catherine University. These courses cover the appropriate content and software development practice, but also include a class project component designed to bolster the course's relevance and appeal. Student evaluations are important both during the course and after its conclusion, and I advocate for my students' professional goals as well. Below are a few of my thoughts on each of these topics.

Adjunct Instruction Experience

My teaching experience is informed by my tenure as an adjunct professor at St. Catherine University. Since 2010, I served as an undergraduate Computer Science instructor in the Math / Physics department. Since 2012, I served as an adjunct technology instructor in the graduate Library and Information Science. I have also worked extensively with the LIS faculty to develop an undergraduate Information Science program, cross-listed with their graduate offerings. I have learned an immense amount about pedagogy by way of my experience.

The student evaluations I've received have given me a good source of average and anonymous feedback. The two independent study courses I've conducted have been with students from previous classes. My experience mentoring them has greatly improved how I approach in-depth mentoring and instruction. During class, I make a point to ask students how they feel about how the course is going as the course progresses. These sources of feedback have proven very valuable for revising course curriculum and my approach in the classroom.

Theory / Practice Cycle

Software engineering is primarily a *practice*. My courses include extensive lab work, assignments, and a class project in three phases (below). Throughout their work, I do my best to introduce and discuss software industry best practices. Those who are interested often incorporate them into their code, others stay with basic requirements; both are fine and satisfy course objectives. The cycle of discussing new Computer Science theory and practicing software engineering skills help each reinforce the other.
From time to time, topics arise which are connected to the course by way of industrial application but which are strictly speaking, out of scope of the course. For instance, data scientists are great consumers of databases but use them as only part of their work. In cases where students ask about topics which are out of class scope, I tell them what I know about it and open the topic for discussion and recommendations for further inquiry. It is important that any topic is allowed in class, as long as the student's inquiry is explored in good faith. Graduate courses and independent study courses have especially benefited from this approach.

Project Orientation

All of my classes include a class project, the goal of which is to make technology development and implementation as relevant as possible for them. Class projects not only reinforce the skills they learn through lab exercises and homework assignments, but it also familiarizes them with the process of brainstorming, designing, developing, debugging, deploying, and using new software.

The class project proceeds in three phases: requirements, design, and implementation. The deliverable for the requirements phase is a requirements document, describing the software project they will undertake (the project must be appropriate to the course; e.g. a database course would require a database project, etc.). The document should provide the context and focus of the software product, describe how it is intended to perform or function, and illustrate some sample output from the software (e.g. data or web page mockups). At this point, students must also specify if they would like to work with others in the class, and if so, whom is part of their "development team".

The design phase entails a deliverable which describes the software in abstract but usable terms; typically graphs. For instance, a database would be described using an entity-relationship diagram such as a Chen diagram or Crow's foot Diagram. Websites would be described with a site map, and wireframe web page mockups would illustrate general design principles (patterns, boilerplate, content areas, etc.). In the implementation phase of the project, students create the software they have described, culminating in a class presentation and demonstration. Successful completion of the class project comprises no less than 30% of their course grade.

Projects are effective at providing students with a toy example or microcosm of a software engineer's career. Students may develop their project across classes (e.g. a website with a database back-end), and often include them as part of their graduation portfolio and job application materials.

Post-Class Advocacy

Over my 20+ year career, I have gotten to know many I.T. job recruiters, some of whom I have entrusted with my own career. If they wish to pursue an I.T. career opportunity, I offer to refer them to a recruiter. If they wish, I also help them prepare for the interview and if successful, the job. As mentioned elsewhere, I have always taught Computer Science to non-majors. Some students decide to pursue a career in software development nonetheless – my own career started this way – and fashion a resume which reinterprets their education and work experience to make it compatible with an I.T. career. I find these students are as successful in I.T. careers as those with CS undergraduate credentials. I am careful to mention this when referring such students to I.T. recruiters.

Student Evaluations

My student evaluations have been consistent with average faculty scores, and are above average for adjunct faculty. Adjunct faculty score overall lower on their evaluations, due mainly to our lack of availability. I use this valuable feedback to refine my course materials and grading criteria. I also try to cultivate relationships with students who are comfortable in their program of study and / or are close to my own age. I ask them personally for feedback, and for their honesty about things about the course which they think are or aren't working for them. I take this feedback seriously as well, though mainly as an indication of how students are feeling rather than performing. Especially for students new to the practice, software development is stressful. It's satisfactory if students are feeling mostly okay with the stress, but not if they feel overwhelmed by it, or overly confused with the process or the course content. Based on this more personal feedback, I may revisit course content for clarification or adjust the pace of the course.

Conclusion

The effectiveness of my teaching style can be evaluated at multiple points, only some of which are reflected in student evaluations. My students' ultimate career success, and especially their comfort in evaluating, developing, and using software, is the best feedback mechanism I know for understanding the success or deficiency of my teaching style. I strive to help develop student competence and confidence, hand in hand. The role of technology in their careers indicates how well I've contributed to their development as a professional.