

Glenn L. Ray, Ph.D.

Teaching Philosophy and Practice

My instructional techniques are guided by:

- Personal experience as a former student and professional IT consultant
- Knowledge and skills needed to successfully compete in a global economy
- Actual student performance in courses that are rigorous and relevant

Every time I teach a course, I learn something and modify my teaching technique accordingly. This is a current snapshot based on teaching information science courses at the University of Pittsburgh.

Teach for long-term success

The overriding goal is preparing students for a successful career in a competitive global society. The focus is on fundamental knowledge to support established professional practices that do not obsolesce and are too valuable for outsourcing.

Do expect effort. Don't assume smarts

Learning requires effort and any reasonably prepared student can succeed in my courses if they make the investment. The necessary level of effort varies, with programming courses requiring the most. Devoting class time to project activities lowers the motivation threshold and flattens the learning curve. Generous extra credit opportunities are offered in exchange for completing good work ahead of schedule.

Achieve ambitious learning objectives

Each syllabus states learning objectives geared toward the deeper learning levels of Bloom's taxonomy (Apply-Analyze-Synthesize-Evaluate). The expected outcome is that a student can apply professional best practices to solving realistic problems.

Integrate learning objectives across courses

It takes a tightly integrated process to build a system but in most IT curricula, courses are discrete and poorly integrated. My students model the analysis, design and implementation phases of a single system in three of my courses building on the case study introduced in the programming textbook. Requirements are traced throughout all phases. Students learn how best practices integrate across the software development process and the critical importance of fixing defects as early as possible.

Establish communication as the fundamental skill

In every course, effective communication is the fundamental objective. Every artifact must communicate with its intended audience. Programming is taught as a form of technical business communication where the audience consists of humans and computers. In my analysis course, requirements are specified in simple, unambiguous natural language using the vocabulary of the business domain without any IT jargon.

Foster a high energy learning environment

Courses ramp up quickly and maintain a brisk pace. Up tempo lectures are limited to one hour with thinking questions to encourage brainstorming. In class activities are linked with lectures so new knowledge can be applied immediately. The workload is challenging and balanced throughout the term.

Learn Actively

Students acquire useful skills through active practice. Classes are hands on with just enough lecture to tackle a realistic problem. The standard for assessing learning outcomes is that if you can't do IT, you haven't learned IT.

Prefer depth over breadth

One of the most common errors is trying to cover too much material in a semester. Survey courses are inevitably restricted to shallow learning levels (Know-Translate-Interpret). As a result, students never acquire advanced skills from solving challenging problems. For example, most CS/IT/IS programs include a survey course in systems analysis and design concepts. At Pitt, I redesigned this into a two-semester sequence: An analysis course (required) and a design course (elective). This innovation creates two substantive courses from one survey course, enabling deeper learning objectives for both.

Follow best practices and use current technologies

For depth-first courses, the critical issue is managing course scope, and professional best practices provide crucial guidance for prioritizing topics. For example, in my analysis-design sequence, students use the Unified Modeling Language and professional object-oriented design principles to develop a suite of high-quality models. In my programming courses, we use the latest version of JAVA and Eclipse, an industry standard IDE. Open source software is used when feasible.

Practice in pairs

Pair programming is a recommended best practice in agile software development and research suggests it pays dividends in the classroom, too. After experimenting with voluntary two-person teams, I have adopted it in all of my courses. Each pair submits several (5-6) deliverables during the term culminating in a final project suitable for professional presentations.

Practice iteratively

In elementary school math, this is repetition. In high school band, it is practice. It's a valuable learning technique in college, too. Students do not develop high quality models in a first draft, and experienced business analysts don't either! The high defect rate in analysis models is a primary motivation for iterative, incremental software development, another best practice. The iterative development of artifacts necessarily requires students to cope with the quality of their own work (aka "eat their own dog food"). Iterative practice is more realistic, yields a higher quality work product, and increases retention.

Mentor intensively

It is not reasonable to expect students to complete challenging assignments without assistance. During in-class work, I am continuously mentoring students. I help them with specific problems and review their work for errors. Much of this work is with individual pairs but it can also be group brainstorming. Mentoring is the key to achieving deeper learning objectives and requires much more energy and insight than lecturing. I find that mentoring in conjunction with iterative practice is the key to effective teaching.

Assess learning individually

Pairs are actively managed throughout the term to maximize each student's learning and accurately measure each student's success. Any student not making a fair contribution is subject to reassignment (either to another teammate or solo). Exams are taken in class without study aids and are heavily weighted toward team-based activities. For each pair, individual exam results are analyzed to gauge relative levels of effort.