Teaching Philosophy

Abhik Roychoudhury
School of Computing, National University of Singapore
abhik@comp.nus.edu.sg

Effective teaching has often been associated with effective dissemination of knowledge leading to advanced understanding of a topic by the students. One can also extend this definition to include the notion of "conceptualization" – where the students not only understand a specific topic of a discipline, but also can extrapolate this understanding in other topics or in real-life problems. As an educator, I feel that effective teaching goes far beyond effective understanding and conceptualization of the students as learners. It involves **integration of the spirit of research into teaching and vice-versa**. What we mean by integration of research into teaching is a situation where the course instructor becomes a *serious learner* himself/herself and the teaching facilitates this process. So, by teaching a course, the instructor is spurred to conduct research on certain cutting edge topics which are later integrated into course(s).

As an example, let me refer to a graduate level module I have offered thrice. The title of this module is "Automated Software Validation". While teaching some traditional formal techniques for software validation, *some of the difficulties in using them for software debugging became clearer to me.* I came to fully realize that the user does not often have a clear specification of the "error" being checked – rather he/she only has an understanding that the behavior of a program for a particular input is unexpected. This inspired me to do research in semi-formal techniques on software fault localization with one of my Ph.D. students. Some of the understanding/experience gained from this effort has been integrated into a lecture in the third offering of the course.

The combined spirit of research by the teacher and students is, by no means, restricted to graduate courses. In the following, I outline some teaching methods which I employed in my *undergraduate teaching* for this purpose. It is worthwhile to mention here that in the last three years I have taught courses on formal verification or logic — topics which are relatively mathematical in nature. Of course an important task in the teaching of such courses is to relate these mathematical topics to the actual practice of computer system design. However, this relationship is often brought out in an extreme way (in textbooks or other teaching materials) — dramatic historical incidents which led to the spectacular failure of computer systems due to the lack of formal verification are presented to the students. This clearly attracts the students' attention, but this interest becomes difficult to retain, possibly because the historical incidents seem far removed. Instead, to motivate more mathematical topics like formal verification, I felt that the students need to *understand* the impact of formal verification in system design; they do *not* need to be surprised. So, in my *undergraduate course* on formal verification of embedded systems, I start with existing practices of verification/validation and how they are intrusive to the design process. I then discuss how a model-based formal approach can help the design cycle, without referring to historic disasters that occurred due to lack of formal verification. This results in a

rather different pedagogical style, where the aim is to discuss the system design cycle with the students rather than surprising the students with the power of formal verification and logic. Once the learners appreciate the differences in the techniques, they develop a more long-lasting interest. This results in a learning process with more active participation from teacher/students — a goal which I always try to achieve in my classes. Furthermore, such classroom discussions have, from time to time, spurred my own research, the results of which have later been integrated into teaching.

As a concrete example, let me again refer to CS 4271, the undergraduate elective offered in the Computer Engineering programme at NUS School of Computing, which I have taught. The title of this module is "Critical Systems and their Verification", and it is meant to focus on formal techniques for design of embedded computing systems. When I started teaching this module, I found a lack of classroom examples through which I could elaborate some practical experiences in validating (substantial) parts of an embedded system. Around the same time, in the context of some research work in modeling reactive embedded systems I have been studying ARM's AMBA system-on-chip bus protocol. Working with a senior undergraduate student (whose final year project I was supervising), I figured that the AMBA protocol had some subtle corner cases leading to a deadlock which are hard to find using informal manual reasoning. This work was published in DATE 2003, a leading conference in embedded system/design automation. More importantly, it provided with a good hands-on case study which I could provide to my students in the CS 4271 course. It serves to illustrate exactly what role automated formal reasoning techniques can hope to play in practice for debugging subtle design errors. I have therefore, included this as a case study in later offerings of the CS4271 course.

In summary, my teaching methods are driven by an effort to cross-fertilize teaching and research. I have adopted these methods in classroom teaching as well as in other forms of teaching such as student supervision. As educators, I feel that **we should strive to break the walls between "teaching" and "research" in our own minds**. This can enhance a new spirit of excellence, allowing us to try and seek the best for our learners.