1 Teaching Summary

Teaching is essential to research because research results, no matter how detailed the description may be, would mean nothing without somebody skilled in the area to interpret what they mean, how they can be enhanced, and how to apply them to benefit the society.

My work on estimating mentor-follower relationships [1] and on the dimensions and growth of developer competencies [2] suggests how the teaching and mentorship could be improved in a research environment as well. The practice and motivation are the two inseparable cornerstones for gaining competence: the motivation ensures more practice and the practice leads to expertise gain and motivates further practice at the next competence area or level. My approach is to allow students to spend sufficient time on practice and, based on what they tend to focus on, to select tasks appropriate for their skill and interests to ensure strong motivational feedback once they succeed. In my experience, the most difficult part was to balance support with the nurturing of independent problem solving, initiative, and creativity.

Formal courses

I do not see many differences among disciplines in terms of basic principles that students need to be taught. Each discipline uses different terminology and within each discipline a variety of fashionable practices with their own buzzword-laden vocabularies are appearing and disappearing over time. Upon closer look, the same principles of rational inquiry, of properly collecting data and inferring conclusions, of creating models of reality capturing aspects salient to the issue at hand are hiding behind the menagerie of terms. Each discipline simply adapts these basic principles to its premises, to the types of problems it addresses, and, in the process, often invents a new vocabulary. In teaching I focus on conveying not just the terminology of a discipline or a practice, but also the endurance and the discipline-invariance of the fundamental principles.

I enjoyed working with more than a dozen graduate and undergraduate students and teaching a number of graduate- and undergraduate-level courses listed below. In addition to the empirical and practice-focused courses I would consider teaching courses in areas that currently lack a firm empirical basis. Teaching such a course would clarify how to strengthen the role of the empirical approach in these domains.

Teaching various topics in data-driven software engineering is a particular challenge, as it requires strong and broad programming skills to acquire, clean, and augment data from operation support systems, good modeling skills needed to select, fit, and interpret suitable statistical, game-theoretic, or machine learning models, and the ability to acquire a non-trivial understanding of practices in the target domain to ensure that the analysis is relevant. I have tried a variety of approaches to address these challenges with the students and for the tutorials. First, I tailor the topics to focus on areas that may not be familiar to the target audience, for example, focus on software data analysis and software engineering practices with students who have a strong computer science background and on programming data-intensive applications with students who have a strong modeling background. Second, I try to build students' knowledge starting from the fundamental elements of measurement and ensuring that students can always see how what they are doing contributes to the ultimate goals. Third, the only way to learn data analysis is by practicing it. I would provide example problems and data from industry and from open source projects to practice data analysis and to evaluate students' progress. Finally, I think it would be necessary to collaborate with educators in the surrounding domains to fine-tune their curricula to aspects important for quantitative software engineering. For example, statistics courses need to be more sensitive to the kinds of data and the types of issues encountered when drawing conclusions from operational support data, where skewed distributions, data missing not at random, and censored observations are not advanced topics but are deeply embedded in any, no matter how simple, data set.

Topics

Vast amounts of data and computing resources can be combined for discovery and innovation. This requires a range of skills from the mathematical modeling and statistics, to programming, data storage and visualization. A deep understanding of human behavior and of underlying technology is also a necessity when analyzing any socio-technical system. In addition to traditional skills and domains listed above, *Digital Archeology* involves a variety of techniques used to clean, integrate, understand, and derive meaningful measures from unstructured data that was never meant to be used for measurement. I would work on educational strategies and courses to fill this broad area.

Statistics in Software Engineering: Pitfalls and Good Practices, ESEC/FSE, Saint Petersburg, Aug, 2013

Measuring globally distributed software development, Pan American Software Quality Institute, July, 2013 http://pasqi.org/events.html

Measuring Globally Distributed Software Development, Pacific Rim Summer School, Beijing, China, 2010, http://www.cpathi18n.org

Measuring Globally Distributed Software Development, Pacific Rim Summer School, Beijing, China, 2010, http://www.cpathi18n.org

Digital Archeology, Mining Software Repositories Summer School, Kingston, ON, 2010.
http://msrcanada.org/school/

Predicting Risk of Software Changes, Mining Software Repositories Summer School, Kingston, ON, 2010.

Empirical Estimates of Software Availability, Mining Software Repositories Summer School, Kingston, ON, 2010.

Chunking Code, Mining Software Repositories Summer School, Kingston, ON, 2010.

Domain-specific defect models, Tsinghua University, Beijing, 2009.

How to run empirical studies using project repositories?, 4th International Advanced School of Empirical Software Engineering, Rio de Janeiro, Brazil 2006.

“Statistical Graphics,” 1994, CMU.

“Spatial Statistics,” 1993, CMU.

“Probability Theory and Random Processes,” 1993 and 1994, CMU.

“Probability and Applied Statistics for Management and Social Sciences,” 1993 and 1994, CMU.

“Probability and Applied Statistics for Physical Sciences,” 1991, CMU.

“Queuing Theory,” 1989 and 1990, CMU.

References

[1] Audris Mockus. Succession: Measuring transfer of code and developer productivity. In *2009 International Conference on Software Engineering*, Vancouver, CA, May 12–22 2009. ACM Press.

[2] Minghui Zhou and Audris Mockus. Developer fluency: Achieving true mastery in software projects. In *ACM SIGSOFT / FSE*, pages 137–146, Santa Fe, New Mexico, November 7–11 2010.