Teaching Statement

Roberto Silveira Silva Filho
http://www.ics.uci.edu/~rsilvafi

My main motivation for applying to a position in academia, as opposed to industry, is the opportunity to teach, advice, inspire and educate both graduate and undergraduate students. I strongly believe in the importance of education in society, and teaching has provided me with rewarding experiences during my tenure at UCI. I believe the knowledge we accumulate as practitioners and researchers is only valuable if shared with others. As such, during my tenure at UCI, I had the opportunity to teach and advice both undergraduate and graduate students.

As a teaching assistant, I lectured discussion sections in classes such as ICS22 (Introduction to Computer Science) and ICS52 (Introduction to Software Engineering). In these classes, I had the opportunity to exercise my teaching philosophy of combining a solid theoretical foundation with practical hands-on experience. As a part of my teaching duties, I also attended a whole quarter on advanced teaching seminars, where I practiced and discussed different teaching issues and course design techniques. As part of this class, also I designed an introductory course on software engineering with emphasis on software design. More than the subject of course design and teaching, the most insightful parts were the discussions with peer young lecturers and the feedback from experienced professors.

As a senior graduate student, I had the opportunity to advice some younger graduate students. Having experienced myself many of the difficulties of a graduate student, for example, the switching from a problem solving to a problem understanding mind set, the pursuit of individual research, and the articulation of results for my dissertation, I've helped many graduate students in overcoming these difficulties. One of the results of my tutoring of younger students was the work on Aspect-Oriented Programming usability, published with Wiwat Ruegmee on the Automated Software Engineering conference in 2008. In this project, I helped Wiwat in framing his research as a usability problem and in designing user studies to further evaluate his dissertation work. I also remotely advised students such as David Bentolila, a M.Sc. student from University of Pará, Brazil, helping in his framing and understanding of the problems surrounding API usability and in the design and development of a usability metrics tool that I have been using in my Ph.D. dissertation.

As a professor, I wish to teach Software Engineering classes and special topic classes such as Topics in Groupware, Middleware and Software Evolution. My approach to teaching is to incorporate both a solid theoretical foundation, with hands-on experience in leading students into thinking about the problem, and formulating hypothesis, before engaging in their solution. The proper teaching of Software Engineering, for example, cannot be limited to the presentation of techniques and notations. Instead, it must focus on the understanding of its essential difficulties, and the principles used in its analysis, design and implementation. This can only be grasped by the combination of theory, as an enlightening factor, with hands-on experiences that allow the students to understand the problem and apply the software engineering principles. I believe many of the problems faced by software engineers in the field are a consequence on the way this subject has been traditionally taught. Traditional software engineering courses focus on the exposition of techniques and approaches without much connection to the problems faced in real-world projects. For example, one of the main differences between programs students develop in school and those one will eventually develop as a professional is that the design problems solved by school programs are not of the complexity and scope of real-world applications. In many software engineering classes, requirements are usually non-ambiguous, they do not change in the middle of the semester, nor new features are introduced when the coding phase is in place. While this is a function of the limited time students can devote to the class (which still need to teach the basic software engineering principles, concepts, tools and approaches), I believe students can and should be exposed to more realistic situations. Hence, one of my goals is to expose the students to as much realistic experiences as possible. For example, In ICS52, Introduction to Software Engineering class, I acted as a client, when I purposely kept some requirements ambiguous or incomplete. The requirements were only revealed gradually, after discussion sessions in the form of query and answer were conducted. Later on in the semester, some requirements were changed, requiring students to modify their documentation and code. I also developed a prototype used by the students to test the requirements of the project. This prototype had a set of bugs introduced as a way to show a more realistic testing scenario. In another class, ICS22 (Introduction to Algorithms), when designing weekly projects or solving a programming problem, I always asked the students to first think about the problem, discuss the requirements, and then start their own interactive design. For such, after this brainstorming phase of some minutes, where the students were given an opportunity to understand the problem, I gave the students some time to solve the problem themselves. Only then I proceeded to guide the students in the solution of the problem at hand. This approach gives the students the opportunity to think about the problem beforehand and experience some of the difficulties one may encounter in real-world projects.

As a long term goal, I would like to design a course on the history of computing, with emphasis on software engineering principles, design successes and failures. I believe in the recurrence of old problems in technology, and the role of history in elucidating common mistakes, good design solutions, and proven principles. For example, many of UNIX design principles are prevailing in current operating systems and general purpose applications. The UNIX minimal core approach, together with its simple yet powerful pipe-and-filter composition model, and the ecology of tools it supports provide lessons to the development of modern computing systems such as component-based software, service-oriented architectures and middleware. Other examples include: the role of modularity in supporting the diversification and proliferation of the hardware and later software industry, as discussed by Baldwin and Clark in their book: “Design Rules”; and failures such as the Therac-25, the Ariane and the Mars Polar Lander, that provide insights on the need for testing, rigor, formalism and specification. Finally, projects such as OS/360 provided invaluable insights on software engineering principles as discussed by Fred Brooks in his book “The Mythical Man Month”. In short, through the systematic study of computing history, its solutions and recurrent problems, one not only have concrete examples of successes and failures in the IT industry, but also is exposed to a set of fundamental design principles and lessons that can be applied to future and existing projects.

In sum, by adopting a teaching philosophy that combines a solid theoretical background with hands-on experience, my goal is to connect theory with practice, thus transmitting the knowledge acquired during years of research first to students, and through them, to society as a whole. When teaching the students, I see myself as a bridge between the research and the practice of software engineering. I recognize my role as a professor as a channel to disseminate the lessons learned in the software engineering research by teaching the leading industry professionals and researches of tomorrow.