Stefano Basagni

Statement of Teaching Philosophy

One of the most vivid memories of my early career as an academic, concerns a conversation with my friend and colleague Chiara Petrioli, now a professor at the University of Roma "La Sapienza." We had just finished polishing the final version of a co-authored paper, which was accepted for publication in one of the most prominent journals in our field of research. We thought we had done a very good job, not just for having the paper accepted, but also in preparing a version for publication that was well-written; clearly organized to highlight our contributions, and poised to become a work that many would have cited and used for further research from then on. On that occasion, I remember Chiara proudly telling me that one of her career goals was to be remembered as somebody who invented something that changed peoples' lives. To that statement, with no hesitation and with the same enthusiasm, I replied: "I would like to be remembered as *the teacher* of somebody who invented something that changed peoples' lives." The spirit of that reply—in fact, the attitude of that reply—has always informed my approach to teaching in all its forms, whether it is preparing lectures for a course or for tutorials; a keynote talk; advising undergraduate and graduate students or writing books and papers about my research discoveries.

Independent of the topic being shared, *teaching* for me is about *enabling the willing listener to desire the acquisition of knowledge; to develop critical and independent thinking and to engage in a life-long journey of discovery of the many facets of truth: self-realization (truth to oneself), advancement (truth in discovery), innovation (new truths) and developing a passion for truth itself (love of truth).*

Specifically, the field of computer engineering provides teachers with the almost unique opportunity of transmitting a passion for learning along with the tools for making peoples' lives better. Independently of the subject matter, computer engineering is about multiple approaches to problem solving, involving a breadth of study and communications skills (particularly important, and a *trade mark* of good education), as well as a strong foundation in the study of computer engineering *big ideas*, such as formal reasoning, abstraction, modularity, complexity and computability, algorithm design techniques, and optimization.

Along with the wide range of computer engineering topics, with their intrinsic beauties, students must also appreciate the depth and power of their immediate usefulness. Computer networking, for instance, is the result of years of brilliant achievements, including algorithmic solutions for shortest path determination; multiple exclusion techniques for making broadcast channel access possible; distributed systems sophistication that guarantees protocol correctness and desired performance. At the same time, however, it allows us to call a loved one anytime and anywhere, to provide critical healthcare information in a timely and secure manner, or to stream the latest movie to our own television, while waiting for the delivery of the groceries we have just ordered online. Enabling the student to make the connection between computer engineering *big ideas*, its specific topics, and their use to solve world problems is the core, and the excitement, of my teaching philosophy and practice.

Methodology. As a teacher, I am a discoverer. The instruments of teaching are the tools of my journey into the world of hearts and minds. The principles that provide a conceptual framework for the goals of my teaching philosophy include the following:

- Encourage students to develop a capacity for self-direction and self-generation of ideas. Learning is a developmental process, which includes being able to formulate questions; provide innovative answers (e.g., research), and being able to both articulate and communicate them (e.g., professional writing). Ultimately, I buy into the idea of the "progressively shrinking professor:" It is when the student does not need me anymore that my job is positively done.
- Stimulate student curiosity and interest, as without engagement there is little motivation to learn. Elegance and beauty of concepts is a compelling starting point, along with promoting active inquiry and critical engagement of real-world problems.
- Experiential learning, i.e., learning by doing. While computer engineering *big ideas* have solid theoretical background, it is by integrating this background and ideas through project work that makes those ideas *concrete* and provides the students with more chances to do original work.
- Encourage student-faculty contact, to provide timely feedback, and to respond to a student specific way of learning. As a teacher, my aim is that of engaging students at their own level, while helping them to stretch and broaden their understating of what is within their grasp.
- Foster cooperation among students through collaborative inquiry (such as team projects where each student has a role, responsibilities and timelines). As learning is also a social activity, a teacher must promote dialogue, envisioning ways in which students can contribute to each other's intellectual growth.

In bringing these principles and approaches to the class, my overarching goal is to develop a student-centered environment. Students should actively participate, feeling empowered to be teachers themselves in a sense.

Evaluation. I provide multiple assessments to determine student comprehension. During my teaching career these assessments have included: Collaborative research projects; formal scholarly research papers; presentations (speaking, multimedia); quizzes and written & oral review activities, as well as traditional exams. In general, the most important part of the evaluation process concerns the discussion of solution recommendations with the students, where we highlight why a specific problem is important, and compare its possible solutions. Although I have tried methods for student self-evaluation (or cross-evaluation, where students evaluate each other's work), I found difficult to maintain general fairness, and have usually resorted to being the final decider of a student class outcome.

Topics I love to teach. In my years as an academic I have taught courses spanning a wide range of topics, all falling under the general umbrella of computer engineering.

I have taught introductory classes on **discrete mathematics**, to provide a basis for the analysis and correctness of algorithms and systems, and for the evaluation of their performance. I have a special fondness for this kind of "foundation" classes, as they allow me to share concepts

highlighting the beauty that can be found and eventually pursued in introducing (and proving) results. I often quote Paul Erdös' "Proofs from *The Book*" stories, and compare different proofs for the same result, not just to show different techniques, but also to emphasize that some proofs are so beautiful that they must belong to *The Book*, where God keeps the perfect proofs of everything. And as Erdös quoted: "You don't have to believe in God, but you should believe in *The Book*."

The same line of thoughts drives my teaching of classes on **algorithms and data structures**, which I have taught for many years and to whose design I have directly contributed. Specifically, the undergraduate version of the course I taught on this topic included programming activities of increasing difficulty, spanning six games/problems, from a game of Master Mind to finding routes on maps. Each game gave me the opportunity to introduce those data structures and algorithm needed to program its solution. The graduate version of this class aims at providing students from many different educational systems a common basis for understanding and using fundamental algorithm design techniques, complexity analysis, comparative performance evaluation and proof of correctness.

With regards to software, I have taught an undergraduate class on **software engineering** and a graduate class on **software construction**. While sharing a common background on basic software management technique, the two classes were designed quite differently. The undergraduate class covered the history of software development and different methodologies for organizing and managing major software undertakings, including company dynamics and interaction with customers. The aim is that of making the students cognizant of the importance of looking at software development from critical and multi-faceted points of view. The class for graduate students aimed at providing the student with a hands-on approach to a general programming language (i.e., C++) and experience in building a medium-size program. The overall goal of the class was to increase those student skills in programming and programming tools that would help them with their individual research projects.

Finally, given that my research concerns networked systems, my major contribution to teaching in recent years has been focused on **computer networking courses**. I have designed classes for both the undergraduate and graduate curriculum. For the undergraduates, I have focused on providing the students (very early in the curriculum) with the fundamental concepts of networking, using the structure of the Internet as a guiding path through the layers of the networking protocol stack. With networking being so pervasive today, and as all our students are already well acquainted with networking applications (as users to begin with), it is quite easy to start with everyday examples, including cell phone uses, social networking "apps," texting, etc. In order to stimulate ideas on possible networking solutions, I will show clips from well-known movies or series, such as Star Trek, involving communication technologies of many types, and begin discussions with the students about what would we need (technology, protocols, etc.) to make that form of communication possible (sometimes this approach to introducing a topic back fires, in that students end up wanting to talk about tele-transport, which is sadly still out of my grasp …). The class has also a programming component, where, by using the python programming language, we introduce network programming via Unix sockets. More important, students are exposed to actual networking experimentation (programming and performance evaluation) through use of NSF GENI, the Global Environment for Network Innovation, a set of networking testbeds distributed throughout the United States that are

accessible via a unified web-based software interface. GENI provides a lab experience without the need for an actual lab space, as it makes several different technologies (wired, wireless, etc.) at several different institutions freely available. Student can perform their experiments from any Internet-connected computer. At the end of the class I usually introduce one or more topics from my own research, which is in networking. These classes usually generate enthusiastic responses from many students, some of whom join my lab for research, often supported by NSF REU grants. For instance, in recent years, I have been supervising several students in a one-on-one fashion through research-oriented courses that the students can take towards their degree (these courses are numbered EECE 4991/2/3 in the current ECE catalog). These courses have led to joint research publications with the students, and to further collaboration.

The graduate courses on computer networking include both basic concepts on networking (especially for those students who did not take classes on this topic in their undergraduate years), as well as more advanced topics. Besides fundamental concepts, these courses include reading and discussing recent papers on various networking technologies, ranging from underwater acoustic and optical networking to terrestrial (radio) wireless sensor networks for environmental and structural monitoring, and software defined networking.

My recent involvement as a founding faculty of the newly minted **Institute for the Wireless Internet of Things** at Northeastern University has brought to the creation of new courses in the general field of wireless networking and of the Internet of things. Two classes that we have been designing within the Institute have been recently approved by the Undergraduate Study Committee and by the Graduate Affairs Committee to be offered starting in Fall 2020 and Spring 2021:

- *EECE 4638. Special Topics in Computer Engineering: Network Architecture Design for the Internet of Things (IoT).* This is a hands-on, lab-type course to provide knowledge of the main differences, challenges and opportunities introduced by the IoT, when compared to traditional computer networks.

- *EECE 5698. Special Topics in Computer Engineering: Network Programming*. This course will provide an introduction on how to develop code libraries and applications that can communicate with each other through modern data networks, using C++ as a programming language.

These two courses are the building blocks of new programs that we intend to create for those students of who are interested in wireless networked systems and the IoT.

In designing all these courses for different audiences, I pay specific attention to the variety of the different cohorts of students, favoring a *tabula rasa* approach with undergraduates while building instead on diverse prerequisites with the graduate students.

Teaching beyond the classroom. Beyond traditional course-based teaching and supervising students in research-oriented courses, I believe it is exciting to actively participate to the department and college-wide discussions about how to organize and implement the ECE undergraduate and graduate curricula. In my years as a member of the ECE Undergraduate

Study Curriculum (that I current co-chair with professor Onabajo) and in the COE Undergraduate Curriculum Committee, I had the unique opportunity and privilege to influence our curricula, and to drive those changes that have kept our programs up to par with accreditation requirements and with the challenges and requirements of our times. More important, I was brought to a deeper appreciation of the new educational opportunities enabled by Northeastern University recent evolution into a global institution. In particular, as a faculty of the ECE department and as a member of the Institute for the Wireless IoT, I plan to promote the following pivotal changes to our undergraduate and graduate offerings.

Innovative educational programs. I will collaborate to design, implement and promote a new minor in Wireless IoT for undergraduates, two new M.S. concentrations and a new experiential Ph.D. program on the themes of wireless networked systems. The two new master's concentrations will be focused on the theory and practice of the wireless spectrum and on the wireless IoT. Programs will be designed to facilitate team teaching of newly developed courses, both through distance learning (synchronously through NUFLEX technology, and recorded). Synergy with Northeastern existing strengths will be key: Ph.D. students will be encouraged to pursue a Leadership Certificate through engaging with stakeholders outside academia while being mentored by their sponsor supervisor and dissertation advisors (LEADERs program). The new M.S. concentrations and experiential Ph.D. will promote coops with non-academic partners, leveraging the Institute industry partners and connections and the hundreds of nation-wide and international industry connections from the Northeastern coop program.

Research-based cooperative education. In close collaboration with the Northeastern coop faculty I intend to explore and refine the scope of traditional coop extending it to include research, internal or external to Northeastern, as an essential component of our curriculum offering. This will require to develop new modules for traditional coop courses such as *EECE 3000 Professional Issues in Engineering* for undergraduate students, and especially *ENCP 6100 Introduction to Cooperative Education* for graduate students. The new modules will particularly address issues concerning research and innovation and multi-institutional collaboration, which are currently missing from current coop education. The resulting new joint industry-academic programs will specifically rely on Northeastern growing network of regional campuses, comprising locations in San Jose, San Francisco, Seattle, and Charlotte. For instance, a student from the Boston campus, while on coop at Intel or Qualcomm in Silicon Valley, will have monthly touch-points with an academic advisor in the San Jose or San Francisco campus. ECE advisors will coordinate with the student's industry supervisor to identify skills sets acquired during coop, create a customized assessment methodology and map the learning to an equivalent ECE Special Topics course. This revolutionary approach will create new incentives for students to pursue practical experiences that counts for curriculum credits and dissertation.

Shareable educational tools. The newly created programs and educational modules will be institutionalized as new undergraduate programs, graduate certificates, Dialogues of Civilization courses, STEM Outreach modules, and as interdisciplinary professional development workshops through the team institutions and delivered across institutions and to research and industry partners. Curricula/modules dissemination plans will be developed to effectively make the new programs available across multiple institutions, especially community colleges and historically underrepresented minority (URM)-based schools.

Industry collaboration and workforce development. It will be easy to leverage the Institute and Northeastern unique access to companies to create workshops and shareable modules to enhance the education and skills of working-age adults and to meet the hiring, retention and talent development needs of employers in the area of wireless networks and IoT. I would love to partner with university key players to design career re-tooling programs that are aligned with Northeastern lifelong learning objectives.