TEACHING STATEMENT

Pooyan Jamshidi (pjamshid@cs.cmu.edu)

During my career (including 7 years as a software engineer and project manager), I have participated in several teaching and mentoring activities. I always find it very rewarding to help others succeed. The opportunity to help students understand the subject matter, encourage them to think deeply about the core insights, and most importantly, excite them to learn is my main motivation for choosing an academic career over industry.

My teaching skills come from a variety of courses that I have taught. I believe, however, that experience alone is not sufficient to deliver a good lecture, and that even the best teachers need to continuously improve their practices. Therefore, I have taken pedagogy classes to learn how to develop new curricula, create active learning environments, write engaging assignments, design fair tests, and evaluate and reflect on a course and its execution to improve it in the future. In addition to getting feedbacks from students, I also ask my colleagues to attend my lectures and give me feedback for improvements. My desire to share my innate passion for learning has led me to teaching, where I can motivate a new generation of scientists and engineers to be as eager to learn.

I have taught a wide set of classes in *software engineering*, *systems*, and *machine learning*: (i) Foundations of Software Engineering (role: guest lecture, course level: advanced undergraduate) at **Carnegie Mellon University**, (ii) SMA: Software Modeling and Analysis (guest lecture, masters) at University of Bern (iii) Cloud Architecture (co-lecturer, masters) at Dublin City University, (iv) Learning in Autonomous Systems (TA, masters) at **Imperial College London**, (v) Software Engineering (lecturer, undergraduate) at Kharazmi University, and (vi) C++ programming/Data Structure (TA, undergraduate) at Amirkabir University of Technology.

1 Teaching

I believe that for computing education to be effective, students must have a high degree of engagement with the course material. Therefore, my goal has always been to make the course **relevant, interesting**, and **fun**. In my experience, the key to achieving these goals is to be **interactive** and **hands-on**, both in class and during office hours. I also incorporate a variety of *active learning* techniques to increase the student engagement [1, 3].

Course Development

I have developed one course from scratch (Software Engineering at Kharazmi University) and helped in developing one course (Cloud Architecture with Claus Pahl). I also took part in a training class at Imperial College where I learned how to create a computer science course: develop the syllabus, homework assignments, choose grading schemes, manage TAs, develop new course extensions and evaluate the course. I then applied for a high-profile teaching certification in the UK, and I am now an Associate Fellow of Higher Education Academy.

Designing project-driven courses [2]. My teaching philosophy revolves around designing interesting projects for students. For instance, in the Software Engineering course at Kharazmi University, I grouped 3-5 students for the course project [2]. They spoke with the stakeholders of software systems, elicited the requirements, and then prepared the design documents. This way of designing a software engineering course gives students the confidence that they can easily design and implement real systems right after they have graduated. A survey conducted independently in the Department of Computer Science at Kharazmi provided the main common written feedback that "students gained a high confidence in software engineering skills" as a result of the software engineering course I taught. As another example, for the Learning in Autonomous Systems course at Imperial College, where I served as a TA, I coded an engine that runs pong games. Students were able to write their reinforcement learning agents on the same platform, and have fun writing agents that are able to compete with one another's. I have always enjoyed teaching of this kind, because students are more engaged with the course.

Designing for student engagement. Relevance is the key to engaging students in learning [3]. For example, not all students will become machine learning developers, but many have applied or will use machine learning concepts such as classification in their study or work. Therefore, I always attempt to relate machine learning concepts to real-world scenarios. I especially talk about my research developing auto-scaling controllers in collaborations with Intel and the challenges we faced developing a reinforcement learning algorithms for

managing cloud resources. As another example, for the Foundation of Software Engineering course at CMU, I designed a guest lecture about architecture practices in companies such as Netflix, Uber, and Spotify (https://goo.gl/w5LNMQ). I noticed that students enjoyed discussions about real practices based on their feedback and learned about tradeoffs involved in the development of real systems.

Class Teaching

Incorporating students' feedback [1]. I believe feedback from students can significantly improve teaching and student engagements. For example, I always create a feedback section into assignments for students to comment on the difficulty of assignments. When I start a course, I also give the students a chance to tell me why they are taking the class. I also ask their prior background in software development and statistics. This helps me tune the level of the class, and specialize my course materials to fit student interests and needs.

Co-teaching (one teach, one support model). After observing in the Foundation of Software Engineering course at CMU (taught by Christian Kästner and Claire Le Goues) how effective co-teaching is, I now strongly believe in co-teaching. Co-teachers can complement one another by discussing the same topic from different perspectives based on their own unique previous experience. The contrast in the dialogue will help students to engage more and learn better.

Artifact-driven teaching. In my experience, students become more confident with respect to their skills throughout a course if they get the opportunity to work with relevant systems instead of toy examples. For example, in a software engineering course that I taught at Kharazmi University, I created teams of 3-5 students that each chose an existing open source project. During the course period, the students got the opportunity to understand the system and its architecture. They identified the best practices and actively talked with other teams to detect common patterns. They also learned about important strategies for increasing the reliability, integration, and performance testing techniques. This gave them an opportunity to become more confident after the course as they have already developed such systems.

Group work teaching [2]. In my experience, software engineering courses can become more effective if students get the opportunity to work together in an environment similar to the real-world software industry. For example, in an undergraduate final project at Imperial College, I asked a team of 7 students to develop a software tool. Students often learn better from their peers [1, 3]. Therefore, I regularly scheduled group meetings and asked for presentations on their progress. By the end of the project, the students had a chance to share a learning experience with all team members in a friendly environment.

Inclusive teaching. I am committed to creating an environment where, irrespective of their background, all students feel equally valued. My goal is to addresses the needs of students with a variety of backgrounds, learning styles, and abilities. Not only I am committed to the ideals of inclusive teaching, but I have also incorporated them into my teaching. When teaching Software Engineering at Kharazmi University in Iran, I took specific steps towards making my classroom a more inclusive place. For instance, I interviewed different colleagues of mine from different background, two of whom were female. My goal was to present the students with a variety of possible career options, especially for female students. We also were able to show a diversity of career paths, including the lead architect, head of research team, backend developer, and requirement engineer. As faculty, I plan to continue on this path, as well as continue to make my classrooms more inclusive by following established research in this area, from such places as CRA-W (http://cra.org/cra-w/), and NCWIT (https://www.ncwit.org/).

Outreach and impact on my teaching [1]. One of the activities that helped me to improve my teaching capabilities was engaging with the public (primary and high school students and non-technical people) to describe my research work. In 2013, I participated in Young Scientists Exhibition in Dublin which is open to all second level students from Ireland. I described my research to pupils in the exhibition. I also was a mentor for a high school (Queens Park School, UK) student for the CREST competition.

Student evaluation. I strongly believe in eliciting student feedback. I incorporate their suggestions, and I get motivation based on their positive comments to continue improving my teaching practices. In my recent guest lecture at CMU, based on the student's feedback, the lecture was "highly interactive," "with lots of relevant and illustrative examples," and "real-world use cases," with "analogies to other subjects." Also, they mentioned that I had "the ability to clearly explain materials based on previous experience in industry" and I gave them "good constructive feedback".

Course Preferences

I am qualified to teach any introductory class on software engineering topics. Given my teaching and research experience, classes that I would be especially well suited for are: *Software Construction, Software Architecture, Cloud Computing and Big Data, Performance-Aware DevOps, and Machine Learning.* I would also like to create new courses related to my research focus, such as: (i) *Advanced machine learning for software engineers*; (ii) *Design patterns for learning-enabled systems.* I am particularly interested in these new courses because I can teach required skills for designing and building learning-enabled systems (to prepare students for Software 2.0 [1]). I would also be very happy to teach advanced graduate courses and seminars related to my research areas.

2 Mentoring

I greatly enjoy mentoring and working closely with students. (**i**) I mentored Changming Xu, an undergraduate student from Washington University in St. Louis, for his **REU** (Research Experiences for Undergraduates) summer project on adversarial machine learning at CMU. After his project, he told me that he had become interested in doing a Ph.D. after graduation, and asked me for a recommendation letter for applying to graduate schools. (**ii**) I also mentored Zhang Haoran on his Masters final project at **Imperial College** on automated configuration testing. After graduation, Zhang joined **Alibaba Group as a senior software engineer** working on a big data system named Petadata. He told me that the experiences of working with me in the research project helped him develop an understanding on performance benchmarking and testing of big data systems. (**iii**) I also mentored Robert Mason and Brian Carroll for their master thesis on auto-scaling in the cloud; both of them are now **lead cloud architects**. Robert informed me recently that the experience of building an infrastructure for cloud auto-scaling on OpenStack platform helped him for integration of cloud solutions with OpenStack. (**iv**) I mentored Yifan Zhai for her thesis on big data application testing, and she now works as a front-end developer. She told me that the experience she gained for her thesis helped her to understand the performance related requirements of customers and design a better user experience.

For **postgraduate research**, I am generally a hands-on mentor and believe that students typically require and expect significant involvement from their mentor. For instance, for a final year student project (7 undergraduate students) at Imperial College, I offered regular meetings to clarify the requirements, then I set up a Github repository to make their collaborative development easier and provided them regular feedbacks [2].

For **undergraduate research**, I would like to be involved in details of progress on daily basis. For instance, I discuss different directions of research with students. I also have daily stand-up meetings to discuss the progress and to check whether there is anything that I can help them to progress better [1]. My mentoring philosophy focuses more on matching the students interests with projects [1]. I believe these are keys to their happiness.

As a mentor, my goal is to help students grow into **independent** and capable researchers. Besides providing technical training, I also want to help students develop important research capabilities and skill sets, including critical thinking, **bigger picture beyond their narrow research**, presentation, and collaboration. I like to foster an open, collaborative environment, which allows me to know them better, communicate with them better, and build trust in the process. These are what I learned from my Ph.D. program (working with Prof. Claus Pahl) and during my second postdoctoral position at CMU (working with Christian Kästner and David Garlan), and I am excited to provide my students with similar experiences to help them succeed in their future careers.

References

[1] Ambrose, Susan A., Michael W. Bridges, Marsha C. Lovett, Michele DiPietro, and Marie K. Norman. "How learning works: 7 research-based principles for smart teaching." 2010.

[2] Oakley, Barbara, Richard M. Felder, Rebecca Brent, and Imad Elhajj. "Turning student groups into effective teams." Journal of student centered learning 2(1): 9-34, 2004.

[3] C. C. Bonwell and J. A. Eison. "Active Learning: Creating Excitement in the Classroom." ASHE-ERIC Higher Education Reports. ERIC, 1991.

[1] https://medium.com/@karpathy/software-2-0-a64152b37c35