Teaching Statement

Morteza Zakeri-Nasrabadi, Ph.D.

December 2023

Towards continual and lifelong learning for all.

"We learn 10 percent of what we read, 20 percent of what we hear, 30 percent of what we see, 50 percent of what we see and hear, 70 percent of what we discuss, 80 percent of what we experience, and 95 percent of what we teach others."

✠ William Glasser (1925—2013)

1 Overview

Teaching is a wonderful activity for me to help students, as well as myself, find a better way of thinking and acting. I want to become a professor because it provides me an opportunity to involve in nurturing the next generation of young talents in my interesting field, *i.e.*, **Computer Science (CS)**. The main challenge for teaching in the CS field is the appropriate management and regular updating of the educational content to cover the rapid changes in the field. Hence, working as a teacher also pushes me to learn more skills and to be an up-to-date researcher in this field. To me, teaching CS courses means not only transmitting existing knowledge to students but also sparking their interest and independence toward lifelong learning. This way the gap between theory and practice in CS education is mitigated systematically. My teaching in CS is centered around governing computational thinking, reverse thinking, decentralized creativity, and lifelong learning. I strongly believe that these factors, especially teaching the ability of lifelong learning, are necessary to prepare young folks for a successful life in the era of artificial intelligence.

2 Teaching Interests

I am qualified and interested to teach *Software Engineering*, *Compiler Design*, *Database Design*, *Algorithm Design*, *Formal Language and Automata*, *Software Architecture*, *Software Methodologies*, *Software Testing*, *Object-Oriented Design*, *Patterns in Software Engineering*, *Artificial Intelligence*, and *Computer Programming*. I am particularly eager to teach undergraduate and graduate-level courses that are directly related to my research projects. Examples of such courses are *Automated and Intelligent Software Engineering*, *Program Analysis and Transformation*, *Clean Software Engineering*, and *Research Methodologies in Software Engineering*. I have designed and prepared syllabi for some of these courses to date.

My past research and teaching experiences cover a wide range of areas in CS, including but not limited to software engineering, software testing, software architecture, compilers, program analysis, machine learning, evolutionary computing, complex networks, natural language processing, data structures and algorithms, databases, and software quality notably testability and security. Given the need, I am eligible and ready to effectively teach courses or hold workshops on the aforementioned subjects.

I also enjoy teaching programming to kids and teenagers, who are eager to learn about computer systems. I had several successful experiences in teaching *Scratch*, *Python*, and *C++* to kids and teenagers. I am planning to design dedicated computing curricula for K—12 students, to cover emerging topics, such as artificial intelligence, computational thinking, data thinking, design thinking, and decentralized creativity, as well as computer programming and application design with the goal of continual and lifelong learning.

3 Teaching Experiences

I taught the *Fundamentals of Compiler Design* course for one semester at the Iran University of Science and Technology (IUST) to B.Sc. students. The website including course materials, designed projects, and source codes is publicly available at *https://m-zakeri.github.io/IUSTCompiler*. During my M.Sc. and Ph.D., I was also a teaching assistant (TA) at the ISUT for about **ten semesters** and in **six different courses**. My primary responsibilities during TA of these courses were designing and grading assignments, programming projects, and exams, holding extra office times, updating course materials, and preparing lecturer slides. In addition, I designed templates for writing assignments, technical reports, and presenting in-class talks, and created several question banks. I was extremely enjoying doing all of these tasks while pursuing my research career.

I worked as a TA for the *Fundamentals of Compiler Design* course lectured by Dr. Saeed Parsa for **nine semesters**. I organized and reviewed **six large compiler projects** (projects' proposals are publicly available on *https://m-zakeri.github.io/IUSTCompiler/projects*), as well as prepared and graded several, quizzes, midterms, and final exams. Moreover, I worked as a TA for two graduate courses *Advanced Compilers (Optimizing Compilers)* and *Advanced Software Engineering*. For both courses, I designed and planned research seminars mainly based on the latest publications in the field to be held by students during the semester.

I volunteered to be a TA for three other graduate courses, *Game Theory*, *Complex Dynamic Networks*, and *Software Architectures*, in which I had already got a very good grade. I prepared a collection of research articles for each course to teach students the state-of-the-art topics corresponding to that course. Developing and teaching online TA classes and group meetings for all mentioned courses in the era of Covid-19 have provided me the opportunity to deal with the specific challenges of all types of teaching environments, mainly classroom and online classes.

4 Teaching Methodology

In my opinion, there is not a single teaching method that suits all types of courses and all students. On the other hand, computer science is a rapidly growing and evolving field. Therefore, the central goal of teaching a subject must be to nurture the ability of students to acquire and learn the subject themselves continuously. To realize this important goal, I typically use a mixed teaching method to provide the best way for conveying knowledge and facilitated the students' learning process. The teaching method I often combine include but are not limited to problem-based learning, project-based learning, presentation-based learning, and flipped classrooms. My teaching method depends on the course topic and subjects, course type *e.g.*, theoretical or laboratory, course level, *i.e.*, graduate or undergraduate, course audiences, and class type, *i.e.*, online or in the classroom. I briefly explain some aspects and tips of my teaching methodology to clarify how it works in practice.

Problem-based learning When teaching a relatively non-trivial and new topic, I ask students trivial questions to involve and engage them in solving an instance problem step by step. For example, in the *Advanced Software Engineering* course I often describe the steps of automated refactoring through a set of simple questions on the 'rename refactoring': Which entity should you refactor? (naming smell detection); Which name should you assign to the entity? (refactoring parameterization); Which parts of the code should be changed? (refactoring propagation); and How should we change these parts (refactoring applications). I found this method very promising since students do not know how to think about solving the problem they first encounter it. I also find it helpful to tell a **brief history of the problem** and the historical path of approaches that have been proposed to address the problem. For example, when I am teaching parsing techniques in the *Compiler Design* course, I begin with the limitation of LL parsing to initiate the LR parsing methods. Soon after, I introduce LR(0) method and its limitations. I then collaborate with students to reach the solutions, such as SLR(1), LR(1), and LALR(1). Meanwhile, I tell students about Dr. Frank DeRemer who is the man behind the practical LR translator. Curious students enjoy learning the topics in this way. Indeed, I found that they need to be motivated well and satisfied with many "Why's" before explaining any "How's" to them.

Project-based learning I use this method to teach the practical sections of courses such as Compiler Design, where there are also many theoretical backgrounds. Students work in groups to build a complete software system, *e.g.*, a compiler front-end. I am a strong proponent of using large open-ended team projects for teaching computer science subjects. There are at least four major benefits of project-based learning. First, working on a large project encourages active and problem-based learning. Students often prefer to collaborate in groups rather than work individually. I have always received highly positive comments regarding learning progress from students who have fished their projects. Secondly, project-based learning help to minimize the gap between theory and practice, highly observed in computer science due to the rapid growth and dynamics of the field. Without strong practical skills, students will face many challenges to enter the industry after graduation. Thirdly, working in groups strengthens students' teamwork skills such as collaboration, communication, ethics, self-learning, and decentralized creativity, all of which are necessary skills for our future society. Lastly, project-based learning allows students to struggle with real-world computational and programming problems, which also appear in their thesis and academic career. Hardworking students often ask me about the possibility to continue working on their projects toward making an industrial product or writing a research paper. The open-ended projects, I designed for the *Compiler Design* course, were used as a basis for defining the theses for three bachelor students.

Presentation-based learning I employed this method in graduate courses when students should know about the state-of-the-art subjects in the field. The students work in pairs, select a first-class journal or conference paper from a set of newly published papers that I provide, and try to give an oral presentation in the class. Other groups ask their questions, and where it is needed I teach a related background to clarify the subject. During the semester, each group is asked to implement or re-execute the method described in the paper, repeat experiments, and where possible improve the approach. This way M.Sc. and Ph.D. students will learn about the latest advances and open problems in the field which help them to find themselves at the center of academic research.

The success of both project-based and presentation-based learning methods requires a solid project management process. The given project should be well-documented, and broken into several step-by-step incremental milestones so that a steady learning curve can be achieved. The course staff should monitor the progress of each team multiple times during the semester. The project also should be novel enough, interesting, and related to the course topic to motivate students to attend it. The project should have a unified framework for fair task assignment and grading but the framework should impose as few technical restrictions as possible to maximize the team utilization. It is better to separate the details of tasks assigned to each group to avoid inter-group plagiarism. The course staff, *i.e.*, both the teacher and TA team should be accessible to every student at dedicated times so that students do not stray too far during the learning and knowledge exploration process. I successfully managed several large projects for *Compiler Design*, *Advanced Software Engineering*, and *Software Architectures* courses when I was a TA at these courses. A majority of groups I mentored, could finish their course project successfully at the end of the semester.

Flipped and online classrooms I found flipped and online classes very useful for teaching software tools and frameworks, such as ANTLR, LEX, YACC, Enterprise Architect, Sci-tool Understand, NetworkX, Scikit-learn, WEKA, etc. In the flipped model, I record my screen while describing the use of software tools. Thereafter, in the classroom, I ask students to work with the introduced tools while doing their assignments and I am ready to answer their questions. In online classes, I share my screen and describe the tool while recording the screen. The student asks their questions in an interactive theme. The advantage of both methods is that the lectures are recorded, and students can watch the lecture's video as many as needed. I received very positive feedback from students about learning to work effectively with a wide range of computer science and software engineering tools and frameworks, as a result of my teaching. These promising results, which occurred during the Covid-19 pandemic, have motivated me to continue online classes further in my teaching. The main challenge of online classes is that the lecturer should be completely prepared and proficient in working with different tools, including the ones used for teaching and online platforms. For this reason, I always practice my teaching session, before the actual class session. Similar challenges exist for flipped classrooms where the course subjects must be taught and recorded offline, which is time and effort-consuming. Nevertheless, I try to regularly update my teaching materials each semester and avoid releasing the same recorded materials.

Evaluation mechanism and grading policy To me, as a teacher, it is important to provide a fair grading framework in which the students are correctly ranked according to their actual effort and learning achievement. I have noticed that, at the same time, grading is one of the most important concerns of many students who attend a course. Therefore, I try to set up a flexible evaluation mechanism and grading policy by dividing the total points into small pieces, continually assessing student outcomes during the semester, and providing feedback and necessary help to each student. In addition to designing a large project for the course, I often provide three or four assignment sets, in-class quizzes, and paper-based exams. I encourage students to participate in volunteer yet graded activities, such as preparing new assignments, projects, course materials, etc. This way they can rectify any missing part of their grade during the semester. For the practical assignments and the course project, I ask each group of students to peer review the work of another group and write a qualitative evaluation report classmates. This way not only all groups are engaged in each other work but also they learn about ethical and fair evaluation. In summary, I do my best to provide fair and stress-free conditions for each individual student who attends my courses.

5 Supervision and Advising Approach

I enjoy helping interesting and hardworking students to become professional and independent researchers. During my Ph.D. at IUST, I was a mentor of more than five M.Sc. theses and three B.Sc. projects. Students come from different cultures with different backgrounds and intend to pursue different career plans for the future. These factors must be considered when choosing a mentoring approach and they often lead to my mentoring and advising approach being specific to every person. One important challenge of advising and mentoring is to establish an effective communication pattern with the students. Holding regular weekly meetings might be effective for some students but unproductive for others. Similarly, detailed scheduling may not be useful for someone while helping others. A decent advising approach should be flexible and customizable according to the student's personality.

Regardless of forming effective personal communication, I always use several best practices to advise students to get things done efficiently and correctly. First of all, I try to familiarize students with systematic research methods such as systematic literature review, describing how to find quality articles in the field, reading and summarizing the papers as quickly as possible, criticizing the papers, and creating a solid research framework based on their needs. The last one emphasizes how to organize different materials, the effective use of tools, such as MS Excel and Python, to facilitate research, and the effective writing of every important note during research. Secondly, I carefully help students to apply the **Work Breakdown Structure (WBS)** method to their research to identify and break the existing works into smaller and doable tasks. I often dedicate considerable time to evaluate the outcome of technical work performed by the students and helping them where necessary as soon as possible.

Another important aspect of the advising process is to consider Diversity, Equity, and Inclusion (DEI) issues. As an advisor, I strive to be aware of DEI issues and take appropriate actions to foster a supportive and respectful environment for my advisees. In terms of my experiences with DEI, I have successfully mentored students from diverse backgrounds and identities during my Ph.D, including both female and male students, as well as Iranian and international students. I have also participated in various workshops and training sessions on DEI topics, such as cultural competence, implicit bias, and inclusive pedagogy.