Idris Hsi
Teaching Statement

Motivation

I consider teaching to be the single most important responsibility that professors have. Advancement in any area of study depends on the ability of its practitioners to transmit knowledge within the field and to future generations. Thus, in my mind, teaching activities cover not only how we transfer knowledge in the classroom but also how we mentor undergraduates to prepare them for future careers in our field, apprentice beginning graduate students to prepare them for future research and teaching, and communicate the results of our research to our community. My interests with teaching have led me to spend some of my graduate career learning and practicing all of these teaching activities and developing lessons and materials to help future graduate students do the same.

History

I have had extensive experience with teaching: supporting existing courses at the College of Computing and the Industrial and Systems Engineering Department and improving the educational infrastructure at the College of Computing.

When I was a teaching assistant for our Introduction to Programming course, required by all Georgia Tech students in the College of Engineering, School of Architecture, and the College of Computing, I developed programs, exams, and led recitation sections. Later, I supervised thirty-five undergraduate teaching assistants as the Head TA. During this time, I co-wrote a manual for Introduction to Programming teaching assistants and designed the administrative infrastructure required to support the large numbers of teaching assistants and students. For these efforts, the College awarded me a Best Graduate Teaching Assistant award.

I have been part of an effort to design a curriculum for a software engineering degree program here at the College. We produced a paper describing our design which is based on a studio model. Software engineering is a difficult subject to teach, mainly because of the inherent complexity and abstract nature of software artifacts and the wide diversity of the activities required to develop them. To convey this complexity and the difficulties with managing development effort, the subject requires a heavy project and practice component. We proposed a virtual software factory that would identify real projects from faculty and industry to provide an actual context for learning software engineering. Eventually, we planned for the results of these projects could be stored in a library to be used as teaching cases in the same way that architecture students evaluate the designs of their peers.

I observed as the College grew during my tenure here that it was no longer reasonable to rely on a small population of senior students to disseminate the various tricks and advice to newer graduate students. I wrote a document that detailed what new PhDs needed to begin doing to ease their transition into the PhD program and have given lectures in the PhD orientation course covering these topics. I discussed activities ranging from beginning a research journal to tracking references using BibTek or EndNote. I covered the activities incidental to research that would be important throughout an academic career, such as networking and developing research objectivity. I also wrote a document that details the process for taking the qualifiers from the written exam to orals. Both documents have been in use for a number of years now and have helped numerous graduate students in the program.

I designed and organized the College's Graduate Teaching Assistant workshop and have taught many of its courses. My interest in training graduate teaching assistants came from observing that many of our PhD students are poorly prepared to perform their duties. For example, teaching assistants often do not provide meaningful positive and negative feedback to students in the process of grading. I included sessions covering the role of grades in instruction and the process of developing grading criteria and feedback mechanisms. A secondary objective I had was to provide training that would complement the existing PhD orientation course by covering necessary graduate student skills, such as designing and presenting talks. The final workshop curriculum covered teaching assistant responsibilities, the basic skills they need to assist professors and students, and the activities required to teach a course, such as presentation design and the construction of assessment materials. During my tenure as its instructor, the GTA training course was the most comprehensive of its kind at Georgia Tech. Some of the course materials that I developed for the class are still in use at the present time by my department.

Currently I am involved with a NSF-funded research effort to study the cognitive and learning practices of students and researchers in emerging interdisciplinary domains, specifically biomedical engineering. Part of my contributions includes a method for characterizing the knowledge elements in their activities to assist the development of a problem-based curriculum for undergraduates in the biomedical engineering degree program here at Georgia Tech. I have developed ethnographic methods to identify what knowledge is required to solve specific

research problems, where they originated from (e.g. biology, mechanical engineering, computer science), whether a problem is solved, unsolved, or unsolvable, and what problems require the development of new techniques. I am currently developing a taxonomy for organizing these knowledge elements. This taxonomy can serve as a foundation for identifying those knowledge elements that are necessary to future bioengineering courses. The overall methodology is applicable to any domain and can be used to assist with the development of future courses.

Philosophy

Not satisfied with classes that relied on rote memorization to judge learning, I developed a model for knowledge acquisition that I have used in the design of both courses and assessment material (e.g. assignments, projects, and exams). Borrowing from the architecture and construction metaphors, I treat basic knowledge, such as facts and mechanical skills, as building materials that form the substance of an area of expertise. Some examples of a building material include equations for calculating permutations, components of a data-flow diagram, or data structures in programming. Frameworks, cognitive, and meta-cognitive skills and techniques compose the tools (e.g. hammers, saws, measuring tape) that allow a person to build structures using these materials. Frameworks that help to structure materials include things such as testing heuristics, program design methods, and domain analysis methods. Basic cognitive skills include memorization, visual reasoning, and analogical reasoning. Meta-cognitive skills are those that allow a person to reflect and analyze their own learning and reasoning abilities and include techniques such as conceptual mapping and goal construction. Naturally, the latter two, cognitive and meta-cognitive skills, while important, are very difficult to teach in a classroom setting although I believe students should be made aware of them.

When I extend this metaphor to the classroom, I consider each subject to have a particular set of materials and tools. A class should deliver the basic materials that will allow students to obtain proficiency in the subject but should emphasize the tools. A student with good tools can always continue to learn about an area after the class has finished. A student with just a collection of materials but few tools will be unable to develop solutions when faced with difficult problems that call for more materials than that student has available. In the best case, a student will learn to create new tools for themselves. I have observed this in teaching programming to new students. Students that simply try to memorize cases and examples of code do poorly relative to students that develop the skills that programming requires. In designing my courses, I ensure that each session and associated assessment material has both fundamental materials that students should learn and the tools that can be used to explain or frame those elements in the larger context of that course.

Future Teaching

I am qualified and interested in teaching both undergraduate and graduate level software engineering, human-computer interaction (with a human-machine systems and cognitive science emphasis), introduction to programming, and ethics. I am also interested in implementing some of the ideas from our curriculum effort – especially building a repository of software engineering projects for design review and having students peer-review designs and code similar to teaching practices in an architecture studio.

My research activities have focused on the area of software maintenance – specifically, program understanding, reverse engineering, and software evolution. My thesis discusses the measurement of conceptual integrity in the user-visible features of software applications. While I will develop seminar courses that cover these topics, I am very interested in developing courses that teach the design of computing applications and information systems with the objective of ensuring conceptual integrity across all system elements, from interface to software architecture to underlying code. I believe that the next generation of practitioners in software engineering will need to maintain the same distinctions that architects and engineers do in the construction of buildings. As component-based technologies and architectures based on design patterns mature and begin to meet international standards for quality and reliability, software engineering will require more practitioners who can design systems that assemble these components into user-level computing features that solve problems for customers. Computing architects will interact with stakeholders, design the computing features that address the goals of the customers, and design the software infrastructure and interfaces that will support those features. Software engineers, in consultation with the designers, will then build and assemble the application according to the design specification. This model will require students to learn a different set of skills and perspectives than those currently taught in software engineering courses to manage these design activities. My research experience in this area, my extensive interdisciplinary background, and my commitment to teaching have prepared me to create and teach courses that serve these objectives.