Teaching Statement
Paul “Will” McBurney

I began teaching at the college level in 2010, and have found it to be very rewarding. I have been the sole instructor for 8 course sections and been a teaching assistant 4 times. My most recent experience comes from teaching Software Engineering (CSE 40232), a senior-level elective course. Using this experience, I have developed certain philosophies on how to teach and manage a course.

Student Feedback

I believe that courses should, as much as possible, have frequent assignments relevant to concurrent lectures, that are graded with an objective, detailed rubric. Having an objective rubric with a clear breakdown of the requirements for each part of the assignment can ensure that all students receive detailed feedback. Some material will require subjectivity, but the grading of the assignments should be as objective and detailed as possible. This is especially important in introductory courses with large class sizes. The reasons for this are two-fold. First, introductory courses form the basis of future understanding in a topic. If the student fails to fully understand an introductory topic, this can have negative consequences throughout the rest of the student’s career. It is vital to identify specific areas that a student needs to improve in. Additionally, in larger classes, which introductory classes tend to be, the instructor’s time must be divided among a large number of students, limiting the opportunities for individual feedback. Detailed, objective rubrics can supplement student understanding.

I have applied detailed rubrics in large class sections while teaching Introduction to Computer Applications (CS 101) at West Virginia University and Discrete Mathematics (CSE 20110) at the University of Notre Dame. In CS 101, I had over 90 students per semester across two sections of the course. To ensure I provided consistent feedback, I kept a rolling list of deductions on each assignment. Every time I found a new error, I would create a new deduction. This ensured every student received the same deduction for the same error. This also allowed me to quickly provide detailed feedback, as I could simply copy-and-paste the reason for the deduction from my rolling list. I applied this strategy to CSE 20110 for both semesters that I TA’d. Students noted that they appreciated the detailed feedback. Additionally, using this detailed rubric allowed me to grade quickly, returning assignments usually by the next class period even in large classes.

Course Projects

In teaching college courses, I believe that projects are a useful tool. This can be especially true in senior-level and graduate level education, where students should learn to manage long-term goals. However, I believe projects should be treated as a series of small deliverables. This allows projects to directly inform class material, rather than be a distraction from it. For example, in my Software Engineering (CSE 40232) class at the University of Notre Dame, I designed deliverables that coincide with lecture materials. In one assignment, students are required to build unit tests
for sections of their source code while we discuss unit testing in class. Related to this, I believe that class assignments, when possible, should encourage students to interact directly with the instructor through out-of-classroom discussions. In this case, the unit tests deliverable can be gone over in a group meeting with the students. Later, in the debugging section of the course, I will alter their code repository, introducing a defect, such that their unit tests can be used to locate the defect. The group meeting allows me to introduce a defect relevant to the unit tests, ensuring the target lesson is learned. Additionally, I can see where there is a gap in unit testing and introduce a defect there which would be harder for the students to find and correct. This both reinforces unit testing as an important tool in software development and as a debugging resource.

Frequent Assignments

Frequent and regular assignment deadlines can be used to detect if a student’s performance declines. If assignment windows are irregular or too infrequent, it becomes difficult to monitor student progress. A decline in performance may be caught too late into the course to correct, and lasting damage may be done. This can result in a student lacking understanding of critical subject matter that they will be expected to understand later. In the classes I have taught, I have ensured students are aware when the next assignment is due. When I have been able to control the course schedule, I have ensured that when one assignment is submitted, the next assignment is assigned. This ensures students remained engaged in the class. To ensure this plan works effectively, I design each assignment to be focused on the course material being covered during the assignment window. After grading each assignment, I evaluate areas that presented the class with the most difficulty, and adjust material covered in class time to address this. An example of applying this comes from my software engineering class. In this class, I have one deliverable scheduled each class week. Because of these assignments, I found that students had difficulty assessing the difference between functional and non-functional requirements. I then briefly revisited the topic in class to clarify the differences with examples from the assignment.

Teaching Interests

I am interested in teaching a variety of graduate and undergraduate courses. I would like to integrate my research interests into graduate classes on software engineering, evolution, and maintenance. At the undergraduate level, I would be most interested in teaching software engineering, discrete mathematics, and programming languages. Additionally, I would be pleased to teach classes on a range other topics, including, but not limited to, introductory programming, data structures, pattern recognition, data mining, artificial intelligence, and algorithms.