

Teaching Statement

Jaime Spacco

1 Background and Philosophy

I'm currently a Visiting Professor of Computer Science at Colgate University, where I'm teaching Computer Networks and Computer Architecture in the Fall 2006 semester. In the Spring 2007 semester, I'll teach CS-1 and advise one or two honors theses. In addition, I am working with students here at Colgate on three research projects.

I have a strong background with Java-based middleware technologies (such as servlets, jsp, object to relational database mapping tools, and so on) and look forward to developing or expanding an upper-level course based on this material. Such a course would be especially useful for students heading into industry after graduation.

I can teach the introductory computer science sequence, programming paradigms, programming languages, networks, computer architecture, introductory databases, network security, and compilers. With some additional preparation, I can teach operating systems or computer graphics.

Practically speaking, I believe that teaching is like anything else worth doing—it takes practice, preparation and hard work to be successful. However, I know that my best qualification is none of these traits—it's passion. For me, teaching is *fun*. In the classroom, I'm exuberant, excited, bursting with enthusiasm... I do whatever it takes to pump energy into the classroom, because I know that students respond to enthusiasm. I like to think of myself as a highly specialized actor, with very high stakes for my performance: When I play my part perfectly, I *truly inspire* the audience in profound ways, and light a lifelong flame of intellectual curiosity that burns with the power to transform the world. I'm an idealist; this type of job satisfaction *matters* to me, deeply. Teaching is my small way of changing the world.

Philosophically, I believe that the problem domain matters when motivating students. Once students have seen a sampling of interesting problems that they recognize from their daily lives, they can more easily approach novel, abstract problems. For example, I teach graph algorithms using the link structure of social networking sites such as www.myspace.com or www.facebook.com; this is an application that virtually all of my students are familiar with and requires little effort on my part to motivate our study of the problem.

1.1 Improving how students learn programming

I want to alleviate the frustration many novices feel when learning to program by providing students with better feedback mechanisms for programming projects and a strong focus on testing and debugging skills. To achieve these goals, I have built Marmoset, an automated

submission and testing system for programming courses. (More information about Marmoset is available in my research statement.) Marmoset provides two key features: a novel type of feedback called *release testing*, and seamless integration of unit-testing and code-coverage.

1.1.1 Release testing

When deciding how much feedback to give students, a common compromise is to split test cases into *public* and *private* tests. Public tests are given to students when the project is assigned and are used during development, while private tests are run against student code after the submission deadline.

Marmoset adds a new type of test called a *release test*. Like private tests, release tests are kept on the server and are not distributed to students. However, unlike with private tests, students can unlock information about the results of release tests by spending a *release token*. For example, students may have 3 release tokens that regenerate every 24 hours; when used, a release token reveals the number of release tests passed and failed and the *names* of the first two release tests failed. Instructors can make the names of release tests as descriptive as desired. Thus, after using a release token for the poker game project, a student may find out that she passed 5 release tests and failed 5 release tests, while the names of the first two failed tests are `testThreeOfAKind` and `testFullHouse`.

Release tests give students a simple reason to start their projects early—the earlier they start, the more release tokens they can use! In addition, the scarcity of release tokens encourages students to think critically about their code before spending a precious token and discourages them from engaging in unproductive programming behaviors, such as repeatedly making small, seemingly random changes to their code in a frantic effort to pass the next test case. Finally, instructors are forced to write their test cases before assigning a project (rather than writing a description and waiting until after the submission deadline to write the test cases), which helps instructors eliminate ambiguities in the project description *before* distributing the project to students.

1.1.2 Test-Driven Development (TDD)

I believe that writing good test cases is an important aspect of computer science that should be emphasized throughout the curriculum. Successful testing is empowering for students because it provides them with practical skills and a rigorous methodology for debugging. Furthermore, the mentality encouraged by testing—carefully considering method pre-conditions and post-conditions and the circumstances under which they may be violated—is similar to constructing a type of empirical proof that a program is correct.

Marmoset supports testing throughout the curriculum by automatically collecting detailed code-coverage information for test suites submitted by each student. Advanced features of Marmoset allow the instructor to reveal additional hints or feedback to students who write better test suites. The added incentive of better feedback helps students directly benefit from writing good test cases.