

CSE 4415 / SWE 5415
Software Testing 2
Fall 2004
Olin Engineering Building, Room 128
Credits: 3.00

SOFTWARE TESTING 2. (Catalog description) Explores structural (glass box) methods for testing software. Testing of variables in simultaneous and sequential combinations, application programmer interfaces, protocols, design by contract, coverage analysis, testability, diagnostics, asserts and other methods to expose errors, regression test frameworks, test-first programming.

Andy Tinkham
Room 238, Olin Engineering
OH: W,Th 2-3
andy@tinkham.org

Cem Kaner
Room 248, Olin Engineering
OH: TBD
kaner@kaner.com

Texts:

- David Astels, *Test Driven Development: A Practical Guide*
- J.B. Rainsberger, *JUnit Recipes*
- Rick Mugridge and Ward Cunningham, *FIT for Developing Software*
- Jim D'Anjou, et al., *The Java Developer's Guide to Eclipse, 2nd Ed.* **OR** David Gallardo, et al., *Eclipse in Action: A Guide for the Java Developer*

Required Tools:

- **Java 5.0**, download from <http://java.sun.com>. This is needed to run Eclipse and the other tools.
- **Eclipse 3.1**, download from <http://www.eclipse.org>. This is a software development environment, including an editor, debugger, and many other features. JUnit integrates well with Eclipse (and is included in the download).
- **JUnit** (included in the Eclipse download). JUnit is an automated testing framework for unit testing in Java.
- **Subclipse**, download from <http://subclipse.tigris.org>. Subclipse is an Eclipse plug-in that allows Eclipse to work with Subversion source control systems, which we'll be using for the code projects this semester.
- **Ruby 1.8.x** programming language, download from <http://www.ruby-lang.org/en/>. Ruby is an increasingly popular scripting language. We'll use it to drive testing of web-based and other applications.
- **Ruby Development Toolkit**, download from <http://sourceforge.net/projects/rubyecclipse>. RDT is an Eclipse plug-in that provides rudimentary support for developing Ruby code in Eclipse

Supplementary Reading:

Note: These readings are completely optional. Some students have found the books helpful, and we recommend you look in these books first if you have questions or want further information about the subjects we cover in class.

- Johannes Link, *Unit Testing in Java: How Tests Drive the Code*
- Kent Beck, *Test Driven Development: By Example*
- Kent Beck, *Extreme Programming Explained: Embrace Change*

- Martin Fowler, *Refactoring*
- Eric M. Burke & Brian M. Coyner, *Java Extreme Programming Cookbook*
- Brian Marick, *The Craft of Software Testing*
- Vincent Massol, *JUnit in Action*
- Laurie Williams and Robert Kessler, *Pair Programming Illuminated*
- Mike Feathers, *Working Effectively with Legacy Code*
- Joshua Kerievsky, *Refactoring to Patterns*
- Mike Cohn, *User Stories Applied for Agile Software Development*
- Jeff Langr, *Agile Java: Crafting Code with Test-Driven Development*

Additional materials:

- We may post additional papers on the blackboard class site, designating some as required reading and others as recommended, as the course progresses. Announcements of these papers will be made in class.

Prerequisites:

- Software Testing 1. (A sound background in black box testing and the logic of test design)
- Java programming.

Grading:

To pass the course, you **MUST** have a passing average on the mid-term test and the final exam.

- **Undergraduates (CSE 4415):** If the average of your mid-term test and your final exam is below 60%, you will fail the course no matter how well you do on the assignments and projects.
- **Graduates (SWE 5415):** If the average of your mid-term test and your final exam is below 70%, you will fail the course no matter how well you do on the assignments and projects.

You can earn grades as follows

- | | |
|---|------|
| ▪ Homework, spot quizzes and in-class assignments | 10% |
| ▪ Projects | 40% |
| ▪ Mid-term test | 20% |
| ▪ Final exam | 30% |
| ▪ Total points available | 100% |

We don't grade on a curve. If everyone gets 90% or more, everyone gets an A. (B is 80-89; C is 70-79; D is 60-69; F is 0-59).

Mid-term Exam: Thursday, October 6, 3:30 p.m.-4:45 p.m.

Final Exam: Must be submitted by Friday, December 16, 10:30 a.m.

Website

We will use Florida Tech's Blackboard website for communication, grade tracking, and possibly submission of assignments. We may also use it to administer some tests and quizzes. Create an account if you don't have one already and sign up for the class. It is your responsibility to provide and maintain accurate contact information on the blackboard site. As we send information out throughout the semester, we will not accept "not getting the email" as an excuse.

Please make sure your name on Blackboard is recognizably similar to your name as listed in the class roll.

Policies

- Spot quizzes and in-class assignments are graded on a one point scale: Either you do well enough to get credit on it, or you don't.
- We will not accept late in-class assignments.
- Homework is graded on a 10-point scale.
- Late homework and projects lose 10% (1 point for homework and 10 points for projects) for each day late. Homework will not be accepted after we have gone over the assignment in class (generally, this will be the next class after the homeworks are due) nor will homeworks or projects be accepted when they are later than 7 days, regardless of when the material is reviewed in class. Scores for homeworks and projects not turned in prior to the acceptance deadlines will be recorded as 0.
- Please prepare all written work on a text editor or word processor. We will accept text files, RTF files, PDF files, Excel files, PowerPoint files, and Microsoft Word format files. If you don't have MS-Office, OpenOffice and other freeware tools can create all of these types of files.
- Submit homework and assignments through Blackboard. For the purpose of assigning late penalty points, we will use the timestamp assigned by Blackboard to determine what time something was submitted. Projects will be submitted through the Subversion version control server. We'll discuss this more in class when the first project is handed out.
- Keep copies of everything you submit. Documents have been lost in the Blackboard submission process.
- **VERY IMPORTANT:** When you submit something to Blackboard, **make sure that the FILE NAME contains: YOUR NAME(S), and the ASSIGNMENT identifier.** As long as it's unique in the class and we can easily recognize it, you can use any combination or truncation of your first and last names or initials. We get a lot of assignments. When we save them to our hard drive, we put them in one directory. When we do that, we no longer know which drop box entry it came from. If we can't identify your file from the file name in that directory, we will cut your assignment grade by 5 points (half a letter grade). You should also make sure that your names and the assignment title are included inside the file. Failure to do this will also result in a 5-point (half a letter grade) reduction for that assignment grade.

Student Collaboration / Academic Integrity:

Homeworks and projects will normally be done by two students working together. You may choose your own pairs, but must work with a different partner for each paired homework and project. Each pair will make a single, joint submission of their work on an assignment. If you find that you are having issues in a particular pairing, you need to come to us as quickly as possible so that we can help you clear up these issues and avoid any negative impacts on your grade.

You may consult other students (both members of this class and students not currently enrolled) when preparing an assignment. Please give credit to each helper by including in the assignment the person's name and a brief description of how he or she helped you.

If you receive help but submit work that fails to acknowledge your helpers, we will zero your paper and may take additional action in accordance with the University's academic integrity policy.

You may not collaborate on the quizzes or exams. The usual rules governing cheating in closed book tests and exams will be applied. Take-home exams (including the final) may not be authored by more than one person. You may consult with other students in the class on the final exam; **HOWEVER**, each person **MUST** submit his or her own work that is obviously developed independently of all other people **AND** all consultants must be acknowledged as directed above.

Course Objectives and Overview

We have four guiding objectives for this course. In our order of priority, they are:

1. You should develop practical competencies in the types of testing a programmer can and should do to her code or that of a peer. Particularly, these types include:

- a. Test-driven programming
 - b. API-level system testing
 - c. Designing powerful & thorough glass-box tests
 - d. Test-driven maintenance of existing code
 - e. Code-level integration testing
2. We designed this course to appeal to employers looking for senior test engineers and test-interested agile programmers. We want the contents of the course and well-done student work products to grab the attention of sophisticated interviewers. .
 3. You should be familiar with current tools designed to support test-driven development.
 4. You should be able to work well in pairs.

In your black box testing course, you learned important general lessons about testing, such as testing from a theory of error, designing tests with an eye to their power (ability to expose problems) and selecting test approaches based on your project context and your objectives. In this course, we apply your knowledge to a new context in which you have the code. You can modify the code to better test or troubleshoot it. We will do a lot of programming in this course due to the focus on the types of testing done by programmers.

Astel's book will be our main text book for this class. The first 8 chapters are informational, and will be covered in lecture. The remaining chapters show an example of TDD. We won't cover this example in class, but we've designed the first project in a way that should let you use this example as a guide. *We strongly suggest that students read & consult this example, particularly those whose Java skills are rusty.*

Lecture Schedule

Week	Topics	Readings	Notes
1 (8/22-8/26)	TDD, Refactorings	Astels, Chapters 1 & 2 Rainsberger, Chapter 1 & 2	8/26: Last day to add
2 (8/29-9/2)	Programming by Intention, JUnit	Astels, Chapters 3 (Skim chapters 4-6 as well) Rainsberger, Chapters 3-5	9/2: Last day to drop without 'W'
3 (9/5-9/9)	JUnit, Mock Objects, TDD of GUIs	Rainsberger, Chapters 6-8 Astels 7-8	
4 (9/12-9/16)	Unit Testing Theory	TBD	
5 (9/19-9/23)	Unit Testing Theory	TBD	
6 (9/26-9/30)	Unit Testing Theory	TBD	
7 (10/3-10/7)	Review & Midterm		10/4: Project 1 due by start of class 10/6: MIDTERM EXAM
8	Midterm & Project 1		10/11: NO CLASS (fall

(10/10-10/14)	Review		break) 10/14: Last day to drop with 'W'
9 (10/17-10/21)	FIT: Developing tables	Mugridge Chapters 1-8 (May want to read example in Chapters 12-19)	
10 (10/24-10/28)	FIT: Developing tables & fixtures	Mugridge Chapters 9-11, 20-24 (May want to read example in Chapters 30-36)	
11 (10/31-11/4)	FIT, Developing fixtures	Mugridge Chapters 25-29	
12 (11/7-11/11)	Ruby		11/10: Project 2 due by start of class
13 (11/14-11/18)	Ruby		
14 (11/21-11/25)	Ruby		11/24: NO CLASS (Thanksgiving)
15 (11/28-12/2)	Project 2 Review, TDD of test tools		
16 (12/5-12/9)	Final project Q&A		12/8: NO CLASS (Study day) – Andy will still have regular office hours
17 (12/12-12/16)			FINALS WEEK 12/16: Final Exam MUST be submitted by 10:30AM