

CS 576 Dependable Software Systems Syllabus

Instructor: Jay Kothari (jayk@drexel.edu)

TA: Maxim Shevertalov (max@drexel.edu)

(Please begin the subject of each email with CS576)

Office Hours: By Appointment

Course Readings: Provided by instructor

Course Website: WebCT

Introduction

Software Testing is a critical element of software quality assurance and represents the ultimate review of a system's source code with the intent of discovering bugs.

Course topics include: formal specification; black box testing; syntax testing; path testing; data-flow testing; mutation testing; program slicing; fault injection and program perturbation; syntax testing; testing within the software development process; test execution automation and test design automation

Prerequisites

This course is intended for graduate students in Software Engineering and Computer Science. Graduate students in other programs may take this course if they have significant programming experience and the permission of the instructor.

Before taking this course, students should be proficient in discrete mathematics, Unix, and programming using C/C++ and Java. Students lacking such a background should take one or both of the following courses before taking CS576:

- Foundations of Computer Science
- Unix Programming Environment

Students seeking to brush up on their Java skills are encouraged to consult the following resources:

- Java Tutorial: <http://www.cs.drexel.edu/~jhk39/teaching/java.ppt>
 - <http://www.javasoft.com>
-

Academic Honesty

As this is a graduate course, there should be no need to discuss the consequences of transgressions to the academic honesty policy of Drexel University, or what constitutes a transgression. If you are unsure of what is acceptable, please consult me *beforehand*. Please read the section on Academic Honesty in the Drexel University Student Handbook if you have not done so already:

<http://www.drexel.edu/studentlife/studenthandbook2002/Judicial/acadhon.html>

By taking this course, you agree to abide by the policy set forth by the university.

Course Setup

The course will consist of weekly lectures, readings, discussions, and assignments which will be posted on the course website. The due dates and details of each assignment, as well as submission instructions will be listed on the course website.

Although this is an online course, it is *not* necessarily self-paced. Course assignments are required to be completed by the specified date. **No late assignments will be accepted**, without previous permission from the instructor. This means, start the assignments early. Unforeseeable circumstances do arise, and will be considered on a case-by-case basis. Late assignments will receive a score of zero. All assignments are considered individual work.

Grading

Letter grades will be given per assignment (A, B, C, D, F). The expected distribution of the course is such that the top 25-35% of students shall receive an A, and the rest shall receive a grade of B. Cs and Fs will not be given unless the student exhibits unusually poor performance.

If you are not satisfied with a grade for a particular assignment you have 48 hours to come discuss the grade for that, and that assignment only with me. Grades are non-negotiable, but in the event that you feel your grade is unjustified I will be more than glad to discuss it with you. Course grades are final.

Course Topics

The major areas we shall cover include:

- General Software Testing: Overview of the maintenance and testing activities within the software life cycle. Syntax Testing. Formal Methods. Black Box Testing Using Formal Methods. Software Related Risks.
- Software Maintenance: Major maintenance activities. Estimating maintenance costs and productivity. Predicting maintainability with software quality metrics. Economics and expectations of software reengineering. Principles of software reuse and reverse engineering techniques.
- Management and Quality Assurance: Cost estimation. Project scheduling. Specification of work units. Quality/complexity metrics. Software availability. Measurement and prediction of software reliability. Software verification, correctness proofs, symbolic execution, walkthroughs, inspections.
- Testing in Small: Testing strategies, including unit level, path and dataflow testing, domain testing, decision tables, and state-based testing. Coverage metrics. Impact of object-oriented testing. Effort, efficiency, and effectiveness concerns. JUnit and JCoverage Testing Tools.
- Testing in Large: Integration (decomposition based, bottom-up, top-down, call graph based, and atomic system functions). Validation and system testing (data, action, port, event and thread testing, structural and functional approaches, operational profiles). Performance and robustness testing.

If you would like to see any other topics covered please let me know and I will work them into the course schedule.

Other

If you have any questions or concerns at all, *please* let me know. I am here to help you get the most you can out of this course and will always be willing to help, so don't hesitate. If there is anything about the course you do not like, or feel could be improved, I will be glad to address it and do my best to accommodate you.

Again, my email address is jayk@drexel.edu. When sending emails to me please place CS576 in the subject field to ensure that I get to them right away.