Spring 2009: EE379K Software Testing
Time/Location: TTh 11am-12:30pm ENS 302A

Instructor

Sarfraz Khurshid
ACES 5.120
x1-8244
`khurshid@ece.utexas.edu`
Office hours: TBD

Catalog entry

Basic concepts and techniques for testing software and finding bugs. Topics include the testing process, unit, integration and system testing, manual and automatic techniques for generation of test inputs and validation of test outputs, and coverage criteria, and focus on functional testing.

Prerequisites

Electrical Engineering 322C with a grade of at least C. Knowledge of Java will be benefical but is not required; language constructs necessary for this course will be introduced in the class. Students must be able to write correct technical English.

Description

This course first introduces the basics of software testing theory and practice, and then presents some recently developed techniques for systematically finding bugs in programs and improving their reliability. A NIST report from 2002 estimates that software failures cost the US economy $59.5 billion dollars annually and over a third of this cost could be saved using a better infrastructure for testing. It is widely accepted that testing currently accounts for more than one half of the cost of software development. Learning the techniques and tools presented in this course is likely to significantly increase the students' productivity as software developers and testers and improve the quality of the code they develop.

Deliverables

There will be four problem sets, a mid-term exam, and a final exam.

Grading

Each problem set will account for 10% of the grade. The mid-term and the final will each account for 30% of the grade.

Textbook—required

Introduction to Software Testing by Paul Amman and Jeff Offutt. Cambridge University Press. ISBN: 0521880386.

Lab resources

Students will need access to a Java development environment. Additional resources may be required based on particular assignments.

Collaboration

Students must solve the problem sets individually. Please be sure to submit your own independent homework solution.

ECE's academic honesty statement

Faculty in the ECE Department are committed to detecting and responding to all instances of scholastic dishonesty and will pursue cases of scholastic dishonesty in accordance with university policy. Scholastic dishonesty, in all its forms, is a blight on our entire academic community. All parties in our community—faculty, staff, and students—are responsible for creating an environment that educates outstanding engineers, and this goal entails excellence in technical skills, self-giving citizenry, an ethical integrity. Industry wants engineers who are competent and fully trustworthy, and both qualities must be developed day by day throughout an entire lifetime. Scholastic dishonesty includes, but is not limited to, cheating, plagiarism, collusion, falsifying academic records, or any act designed to give an unfair academic advantage to the student. The fact that you are in this class as an engineering student is testament to your abilities. Penalties for scholastic dishonesty are severe and can include, but are not limited to, a written reprimand, a zero on the assignment/exam, re-taking the exam in question, an F in the course, or expulsion from the University. Don't jeopardize your career by an act of scholastic dishonesty. Details about academic integrity and what constitutes scholastic dishonesty can be found at the website for the UT Dean of Students Office and the General Information Catalog, Section 11-802.

Calendar (tentative)

Week 1	1/20	Introduction and course overview
	1/22	Java and JUnit basics
Week 2	1/27	Graph theory, logic, and discrete math basics
	1/29	Chapter 1: Basic software testing principles and concepts
Week 3	2/ 3	Chapter 2: Graph coverage
		Graph coverage criteria
	2/ 5	Chapter 2: Graph coverage
		Graph coverage of designs/specifications/use-cases
Week 4	2/10	Chapter 3: Logic coverage
		Logic expression coverage criteria
	2/12	Chapter 3: Logic coverage
		Logic coverage of specifications/finite-state machines
Week 5	2/17	Chapter 4: Input space partitioning
		Input domain modeling
	2/19	Chapter 4: Input space partitioning
		Combination strategies criteria and constraints among partitions
Week 6	2/26	Chapter 5: Syntax-based Testing
		Syntax-based coverage
	2/28	Chapter 5: Syntax-based Testing
		Specification-based and input space grammars
Week 7	3/ 3	Chapter 6: Practical considerations
		Regression testing
	3/ 5	Chapter 6: Practical considerations
		Test process and test plans
Week 8	3/10	Chapter 7: Engineering criteria for technologies
		Testing object-oriented software and web applications
	3/12	Mid-term exam
Week 9		Spring break
Week 10	3/24	Chapter 7: Engineering criteria for technologies
		Testing GUIs and real-time/embedded software
	3/26	Chapter 8: Building testing tools
		Instrumentation
Week 11	3/31	Symbolic execution
	4/ 2	Combinatorial testing
Week 12	4/ 7	Constraint-based testing
	4/ 9	Parallel algorithms for testing
Week 13	4/14	Incremental algorithms for testing
	4/16	Stateful model checking
Week 14	4/21	Stateless model checking
	4/23	Bounded model checking
Week 15	4/28	Directed/heuristic model checking
	4/30	TBD
Week 16	5/ 5	TBD
	5/ 7	Review