

CEN 6070
Software Testing and Verification

Fall 2008

Overview: Software Testing and Verification is an advanced survey course on concepts, principles, and techniques related to software testing and formal program verification. Students will become acquainted with both the strengths and limitations of various functional and structural test design methods, as well as techniques for proving the functional correctness of sequential programs. Topics include: black-box and white-box test design strategies, incremental integration testing techniques, inspections and reviews, axiomatic verification techniques, predicate transforms, and function-based verification. Students will have the opportunity to practice the techniques presented in class via optional exercises.

Prerequisites:

- (1) Successful completion of an upper division undergraduate or graduate-level software engineering survey course (such as CEN 3031/5035), or comparable professional experience. (Off-campus EDGE students currently employed as software professionals meet this requirement.)
 - (2) Familiarity with programming using a high-level language (C, C++, Java, etc.), and
 - (3) Basic knowledge of algorithms, data structures, object-oriented design principles, and discrete math.
- Note:** item (1) will be strictly enforced. If you do not meet this prerequisite, please plan to enroll in CEN 5035 in Spring09 and then register for CEN 6070 in Summer09. See the instructor for additional information.

Textbook: A collection of *required* readings will be available for purchase as a packet. An optional textbook, Pezze and Young's *Software Testing and Analysis*, Wiley, 2008, is recommended for students who wish to have additional software testing and analysis reference material at their disposal.

Outline of Course Topics:

Intro to V&V Techniques and Principles	Formal Program Specification
Requirements and Specifications	Axiomatic Verification
Black-Box Test Case Design Strategies	Weak Correctness
Partition Testing	Rules of Inference
Combinatorial Approaches	Strong Correctness
Other Strategies	Predicate Transforms
White-Box Test Case Design Strategies	Proving Strong Correctness
Logic Coverage	Computing Weakest Pre-conditions
Dataflow Coverage	Functional Verification
Path Conditions and Symbolic Evaluation	Complete and Sufficient Correctness
Other Strategies	Axiom of Replacement
Integration and Higher Level Testing	Correctness Conditions
Testing Object-Oriented Software	Iteration Recursion Lemma
Reviews and Inspections	Revisiting Loop Invariants
Testing Tools	Cleanroom Software Engineering

Examinations and Grades: Course grades will be based *SOLELY* on two equally weighted 90-minute exams.

Workload: Probably nominal for a non-programming upper division/graduate level course. *Students who are conscientious in completing optional exercises tend to perform much better on the exams.*