Spring 2016: EE382C.3 Verification and Validation

Instructor

Sarfraz Khurshid
ACES 5.120
(512)-471-8244
khurshid@ece.utexas.edu
Office hours: by appointment

Teaching Assistant

Allison Sullivan
allisonksullivan@gmail.com
Office hours: by appointment

Time and Location

All lectures will be 1:00-5:00pm at Commons Building at PRC (PRC 1.130).

Prerequisites

The students are expected to have basic knowledge of discrete math (e.g., sets, relations and propositional logic), data structures (e.g., linked-lists and binary trees) and object-oriented programming (preferably in Java, C++, or C#) and considerable programming experience.

Catalog entry

Advanced concepts and techniques for checking the correctness of programs and specifications. Topics include state-of-the-art research in systematic software testing, state-space exploration techniques, including symbolic techniques, for software model checking, heuristics-based approaches, as well as static program analyses, including those based on automated theorem proving, and behavioral specification languages.
Description

The process of software validation includes reasoning about (the correctness of) programs, whether formally—a process that is termed verification—or informally, and testing programs. This course focuses on verification and testing. 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.

The course is organized as a series of lectures on basics of software testing as well as more advanced research/tool papers. The selected papers will cover traditional and state-of-the-art techniques for software testing and verification. (An initial list of candidate papers will be provided at the beginning of class. Different papers may be selected in view of class preferences.) The course content will cover both techniques for dynamic analysis, such as glass box and black box testing, equivalence partitioning, boundary value analysis, test strategy and automation, regression testing and debugging, techniques for static analysis, and also techniques for software model checking including those that employ heuristics based on artificial intelligence.

Grading

The grade will be based on class participation (10%), homeworks (20%), exams (40%), and a final group project (30%). Students must participate actively in the class. The final project will be done in a group of two or three students. A typical project would involve performing a case study using some tool(s) studied in the class. With instructor’s permission, the students may choose to work on a suitable idea of their own. Good projects will result in work that is of a quality expected for conference/workshop publication. At the end of the course, students will present their projects to the class.

Textbooks (Recommended)

The following texts provide some basic material that would help students understand the more advanced material in the papers:

1. *Introduction to Software Testing* by Paul Amman and Jeff Offutt. ISBN: 0521880386


Calendar (tentative)

Note the following tentative schedule for the problem sets and exams:

Problem Set 1
  OUT: January 16; DUE: February 10

Problem Set 2 [Project Proposal]
  OUT: January 16; DUE: February 28

Problem Set 3
  OUT: February 6; DUE: March 9

Problem Set 4 [Project Update]
  OUT: March 1; DUE: March 31

Problem Set 5 [Final Project Report]
  OUT: April 1; DUE: April 30

Exam 1: March 5

Exam 2: April 2

References

type-oriented programs with invariants. In Proc. ECOOP 2003 Workshop on Formal

Model-based testing with asml.net. In 1st European Conference on Model-Driven Soft-


from counterexamples. In Proc. 26th International Conference on Software Engineering


