A Unified Approach to Verification and Validation of Software Systems
CS378 – Fall 2008
Unique Number – 55785
TTH 9:30AM-11AM RLM 6.126
http://www.cs.utexas.edu/~browne/uvvf2008/

J.C Browne and W. Hunt
browne@cs.utexas.edu – hunt@cs.utexas.edu

1. Motivation and Goal

Correctness is the most critical concern of the software industry. Computers are increasingly assuming central roles in safety- and security-critical systems, leading to dire consequences of viruses, worms, and software faults. Almost all of these viruses, security attacks, and equipment malfunctions are due to flaws in software design and implementation that could have been found by a truly comprehensive and well-structured process to verify and validate the properties and behaviors of the software. Additionally, there are specifications for information flow, which are sometimes called security policies, and the design and implementation of these security policies also must be verified and validated. The methods needed to verify and validate the security policies largely overlap with those needed to verify and validate other types of specifications.

The goal of this course is to make available to students in Computer Sciences at the University of Texas unique training in verification and validation across functional, security and performance properties. Students successfully completing this course will find themselves with a unique, highly valuable and saleable skill. They will also be part of an NSF project for developing and applying a unified approach to verification and validation of software systems. They will also work with state of the art tools and methods for verification and validation.

The class will also satisfy the substantial writing component requirement and will give an opportunity to practice and develop presentation skills.

1.1 Background

The methods and tools which are available for validating and verifying software includes static analysis of program code, conventional and systematic testing, model checking for temporal properties, runtime monitoring, and formal proofs of correctness. Yet there does not exist a unified approach to verification and validation which integrates the several methods and tools for verification and validation. Teaching of verification and validation reflects this fragmentation. This course is part of an effort to provide such a unified and integrated approach. The instructor and Profess Calvin Lin have obtained funding from the National Science Foundation to develop a unified approach to verification and validation and an undergraduate course teaching this unified approach.

The two unifying concepts are a “universal” property specification language from which properties can be verified by static analysis, testing, model checking, proof methods or compiled to runtime monitors as appropriate or required and the insight that all methods of verification and validation are searches of the state space of a program for truth or falsity of specified properties. A third unifying conceptual element is the common set of component-oriented set of design principles which enable effective and scalable application of both formal and informal validation and verification.

1.2 Course Content

The lectures will cover the principles and methods. The participants in the course will follow an example through the steps in an integrated process. They will also evaluate the tools which are available for each aspect of the method. Participants will come away from this course with a unique perspective on verification and validation.

The principles and mechanisms for validation and verification are language independent but the tools implementing the mechanisms are language specific. The lectures will be largely language independent but the examples and the outside assignments will use Java and C. A substantial portion of the lectures will be devoted to design for verification and validation and an integrated and comprehensive approach to specification of properties to be verified and evaluated.

The content for the course will include:

a. Design for test and verification.
b. Unified Property Specification
c. Introduction to program analysis (static analysis methods).
d. Formal and complete approaches to testing:
    Specification of properties, behaviors and assertion
    Test coverage algorithms based on static analysis processes
    Testing as a continuous process integrating runtime monitoring with conventional testing, model checking and proof-based verification.
e. Applied model checking:
    Model checking as the endpoint of testing
    Property formulation
    Compositional reasoning
f. Classical Dijkstra/Hoare and other proof-based verification.
    This material is already covered in other courses and will not be repeated but the role of this material in a comprehensive approach to verification and validation will be covered.
g. Run-Time Monitoring
    Methods and Tools
    Automated compilation of property monitors.
h. Integration of all the methods in a coherent, complete structure for validation and verification.

i. Extension of verification and validation to security policy issues such as information flow.
j. Failure analysis, fault-tolerance, practical self-stabilization, etc.
k. Verification and validation of non-functional properties such as performance.

2. Student Prerequisites

Upper division standing. CS 336, CS 337 and CS 375 are desirable. Students may wish to consult with the instructor either by email (browne@cs.utexas.edu), by telephone (471-9579) or in person before registering for this course.

3. Texts and Course Materials

The text for this course is “Software Testing and Analysis” by Pezze and Young. There are many monographs and texts focusing on each topic concerning validation and verification (particularly testing). There are survey and tutorial articles and a large amount of web-based material is available on each topic and these will be used in the class.

4. Course Work and Grading

This is mainly a project course but there will be a single examination about two-thirds of the way through the semester. Projects may be individual or small team. Writing and presentations on the project are a key part of the course. There will be three progress reports on each project. The first progress report will be a detailed specification of the project. The second progress report will be a specification of the properties to be verified and the approach to development of the system. The third progress report will describe the implementation of the system and a first report on verification results. The final report will evaluate the tools and methods uses and the results obtained. The will be two presentations on each project. The first presentation will cover the system the material in the first and second progress reports and the second presentation will be on the third progress report and the preliminary results on verification. Grades will be assigned on the basis of the presentation, the report and content of the project and will be based two-thirds on the project, the reports and presentations and one third on the examination.

5. Approximate Lecture Schedule

An approximate lecture schedule follows. The time allocated for each topic may vary. There will be several guest lectures by experts on some of the topics.

Lecture Date	Lecture Topic	Reference Material
8/28/2008	Unified Approach to Verification and Validation	Lecture Notes

9/2/2008	Designing for Verification and Validation	Lecture Notes and web references
9/4/2008	Property Specification : Temporal Logics, Floyd/Hoare Logics, JML Pre-conditions, Post-conditions, invariants, etc	Lecture Notes and Web references: http://cnx.org/content/m12317/latest/ http://ieeexplore.ieee.org/iel5/6783/18169/00841031.pdf ftp://ftp.cs.iastate.edu/pub/leavens/JML/jmldbc.pdf
9/9/2008	Property Specification : Temporal Logics, Floyd/Hoare Logics, JML Pre-conditions, Post-conditions, invariants, etc	Lecture Notes and Web references: http://cnx.org/content/m12317/latest/ http://ieeexplore.ieee.org/iel5/6783/18169/00841031.pdf ftp://ftp.cs.iastate.edu/pub/leavens/JML/jmldbc.pdf
9/11/2008	Models, Abstractions and Compositionality	PY – Chapters 2, 3 and 5, lecture notes
9/16/2008	Static Analysis – Type checking, data flow analysis, control flow analysis	Lecture Notes, PY – Chapters 5, 6 and 13. and web references
9/18/2008	Testing – Transition from informal to structured testing	PY Chapters 5 and 6, lecture notes and web references
9/23/2008	Fundamentals of Model Checking	Lecture Notes, PY – Chapters 5 and 8, web references.
9/25/2008	Symbolic Execution and its applications	Lecture Notes , PY – Chapter 7 and web references
9/30/2008	Translation/Abstraction based Unification of Static Analysis, Testing, Model Checking and Runtime Monitoring	Lecture Notes and web references Lecture Notes, PY –Chapter 19 and web references: http://portal.acm.org/citation.cfm?id=760066
10/02/2008	Testing in Depth	PY – Chapters 10,11,12,13,14,15
10/07/2008	Fundamentals of Proof Methods	Lecture Notes, PY – Chapters 5 and 8, web references.
10/9/2008	Automated Proof Systems - Demonstrations	(Systems to be chosen) Key, etc.
10/14/2008	Proof Systems – Software Case Study	Lecture Notes and web references
10/16/2008	Mid-Term Exam – CS378 Section	
10/22/2008	Proof Methods Revisited	Lecture Notes and web references
10/24/2008	Automated Formal Proof Methods	Guest Lecture
10/29/2008	Model Checking – Case Studies	
10/31/2008	Specification and Verification of Non-functional Properties – Performance and Security	Lecture notes and web materials

11/04/2008	Process Algebras and Process Calculi	Lecture Notes and web references
11/06/2008	Guest Lecture	
11/11/2008	Guest Lecture	
11/13/2008	Project Presentations	
11/18/2008	Project Presentations	
11/20/2008	Project Presentations	
11/25/2008	Project Presentations	
12/2/2008	Project Presentations	
12/4/2008	Project Presentations	