**Basic Course Information:** CS 5850

Course Title: Software Verification and Validation

Units: 3 units

CS number: C-2

Component: Lecture

Instructional Mode: Face-to-Face and web-assisted

Grading Basis: Graded only

Repeated Basis: May be taken only once

Cross listed Course: N/A

Dual-listed Course: N/A

Major course/Service course/GE course: Major course

Date Prepared: March 31, 2015

Prepared by: Salam Salloum and Yu Sun

# I. Catalog Description

Techniques for evaluating software quality and integrity. Quality assessment, proof of correctness, testing methods and tools. Survey of trending researches in software testing and verification.

**II. Required Coursework and Background**

Pre-requisite(s): CS 4800 or consent of instructor.

# III. Expected Outcomes

On successful completion of this course, students will be able to:

1. Appreciate the role of validation and verification methods in the system life cycle; key issues in software testing, testing levels and testing techniques.
2. Adhere to software quality engineering principles.
3. Recognize the importance of adhering to software engineering principles of validation and verification in the design and development of test methods.
4. Gain experience in inspection and debugging approaches, configuration management, performance, and quality standards issues.
5. Apply validation and verification methods to ensure and improve quality of software systems.
6. Measure and evaluate a software system's quality, complexity and reliability using software metrics.
7. Use industry-strength IDEs such as Eclipse, NetBeans and IntelliJ and unit testing with JUnit, code coverage tools such as Cobertura, and other similar products to help detect software system defects.
8. Conduct continuous integration (CI) at unit, integration and system testing level using a CI server such as Jenkins to automatically run regression test suites on the system under test.
9. Analyze and control defects in complex systems.
10. Be familiar with the common validation and verification techniques for mobile and web platforms

Outcomes of this course will build student capacity in each of the following areas as defined by programmatic objectives for the computer science major.

P-SLO 3. An ability to build applications, either individually or in a team, that are robust, reliable, and maintainable.

P-SLO 4. A breadth of advanced knowledge and skills in applied areas of computer science.

# IV. Instructional Materials

Text:

1. P. Amman, J. Offutt, “Introduction to Software Testing”, Cambridge University Press, New York, NY, 2008. ISBN-978-0-521-88038-1.

References:

1. Yue Jia and Mark Harman, "An Analysis and Survey of the Development of Mutation Testing", IEEE Transactions on Software Engineering, vol. 37 no. 5, pp. 649 – 678, September 2011.
2. Shin Yoo and Mark Harman, "Regression Testing Minimisation, Selection and Prioritisation: A Survey", Journal of Software Testing, Verification and Reliability, 22(2):67-120.
3. S. Anand, E. Burke, T. Y. Chen, J. Clark, M. B. Cohen, W. Grieskamp, M. Harman, M. J. Harrold and P. McMinn., "An Orchestrated Survey on Automated Software Test Case Generation", Journal of Systems and Software, vol. 86, no. 8, August 2013, pp. 1978–2001.
4. P. McMinn, "Search-Based Software Test Data Generation: A Survey", Software Testing, Verification and Reliability, vol. 14, no. 2, June 2004, pp. 105–156.
5. Alex Groce, Amin Alipour, and Rahul Gopinath, "Coverage and its discontents", Wayward Essay 2014

# V. Minimum Student Material

Textbook and class handouts

# VI. Minimum College Facilities

A classroom with a projection system, a computer laboratory

# VII. Course Outline

1. Introduction to software validation and verifcation
2. Kinds of testing (unit, functional, integration, system, acceptance, regression)
3. Black Box Testing: input partitioning and combinatorial testing
4. Coverage and Structural Testing
5. Mutation Testing
6. Regression Testing
7. Specifying and verifying programs
8. Hoare Logic
9. Model Checking
10. Mobile testing
11. Web testing
12. Mining-based software testing
13. Software performance verification and validation
14. Software testing and verification in industry

# VIII. Instructional Methods

Lecture

Problem-solving/Discussion

In-class exercises

Small group activities

Project-based learning

# IX. Evaluation of Outcomes

A. Student Assessment

1. Homework assignments

2. Projects

3. Exams

4. Presentation

B. Meaningful Writing Assignment

Students shall produce written solutions that are assigned as homework.

Students are required to write commentaries to major program components of each programming projects.

C. A Matrix of Course Student Learning Outcomes vs Methods of Assessment

If the course is being evaluated for accreditation purposes, approved department accreditation assessment tools will additionally be utilized.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Course Learning Outcomes | Methods of Assessment | | | |
| Homework | Programming Projects | Presentation | Exams |
| Appreciate the role of validation and verification methods in the system life cycle; key issues in software testing, testing levels and testing techniques. | x |  |  | X |
| Adhere to software quality engineering principles. | x |  |  | X |
| Recognize the importance of adhering to software engineering principles of validation and verification in the design and development of test methods. | x |  | x | X |
| Gain experience in inspection and debugging approaches, configuration management, performance, and quality standards issues. | x | X |  | X |
| Apply validation and verification methods to ensure and improve quality of software systems. | x | X | x | X |
| Measure and evaluate a software system's quality, complexity and reliability using software metrics. | x | X |  | X |
| Use industry-strength IDEs such as Eclipse, NetBeans and IntelliJ and unit testing with JUnit, code coverage tools such as Cobertura, and other similar products to help detect software system defects. | X | X |  | X |
| Conduct continuous integration (CI) at unit, integration and system testing level using a CI server such as Jenkins to automatically run regression test suites on the system under test. | X | X |  | X |
| Analyze and control defects in complex systems | X | X |  | X |
| Appreciate the role of validation and verification methods in the system life cycle; key issues in software testing, testing levels and testing techniques. | X | X | X | X |