SE2831 Introduction to Software Verification

Dr. Walter Schilling

Fall, 2011

You may use one (1) 8.5 x 11 sheet of paper with notes on it for the exam.

1. Week #1
   - (a) Lecture #1
     - i. No Class due to Labor day.
   - (b) Lecture #2 Introduction to Software Testing
     - i. Explain the relationship between the cost of fixing a defect and the phase in which the defect is discovered.
     - ii. Justify the importance of software testing from an economic standpoint.
     - iii. Explain through case studies the root cause of one or more software failures.
2. Week #2
   - (a) Lecture #1 An Overview of Software Development Lifecycles
     - i. List and explain Polyas four principles to problem solving.
     - ii. Draw the Waterfall model for software development
     - iii. Explain the purpose for each step within the waterfall process
     - iv. Draw a representation of an incremental model of software development
     - v. List the advantages of an incremental model over a waterfall model
     - vi. Compare and contrast the incremental model and the waterfall model
     - vii. Explain and draw the spiral model for software development
     - viii. Draw the V Model for software development
     - ix. Explain the roles of testing within the V Model
   - (b) Lecture #2 Your First Unit Tests
     - i. Define testing.
     - ii. Define the relationship between errors, defects, and failures
     - iii. Compare and Contrast the four main levels of testing.
     - iv. Explain why it is impossible to test every case of program execution.
     - v. Construct rudimentary test case for a simple software method.
3. Week #3
   - (a) Lecture #1 Equivalence Class Testing
     - i. Define the terms black box test and white box test.
     - ii. Explain the concept of a test oracle.
     - iii. Define equivalence class.
     - iv. Explain the concept of design by contract. (Reading only)
     - v. Compare and contrast defensive testing and testing by contract. (Reading only)
     - vi. Given a software description, define the equivalence classes for a given problem.
     - vii. Given a software description, construct test cases using the equivalence partitioning method.
     - viii. Based on Equivalence class testing, determine the minimum number of test cases necessary to test a given software system.
   - (b) Lecture #2 Boundary Value Testing
     - i. Define a software boundary condition.
     - ii. Explain why boundary conditions represent commonly occurring mistakes.

iii. Given a software description, construct test cases using the boundary value testing method.

iv. Compare and contrast boundary value testing with equivalence class testing.

v. Based on boundary value testing, determine the minimum number of tests required to test a given software system.

vi. Explain how one can combine boundary value testing with equivalence class testing to solve multi-dimensional data sets.

vii. Construct test cases which would allow for the testing of multi-dimensional data sets.

4. Week #4

(a) Lecture #1 Automating your unit tests with JUnit.

i. Explain the purpose for the JUnit framework.

ii. Draw the architecture for the JUnit framework.

iii. Define the JUnit terms test fixture, unit test, test case, test suite, and test runner.

iv. Draw the initialization flow for JUnit when executing tests.

v. Critique the limitations of JUnit.

vi. Construct a rudimentary unit test using JUNit.

(b) Lecture #2 State Transition Testing

i. Define the terms state, transition, event, and action.

ii. Explain the concept of a state transition table.

iii. Given a state diagram, construct a state transition table for the problem.

iv. Explain how state transition test cases can be constructed from a state machine definition.

v. Construct a set of test cases from a state diagram.

5. Week #5

(a) Lecture #1

i. Catch up on anything not yet covered.

ii. Review materials for the exam.

(b) Lecture #2 Midterm Exam

i. Successfully master the exam material presented thusfar.