SE2832 Introduction to Software Verification

Dr. Walter Schilling

Spring, 2013-2014

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 Introduction to Software Failure
      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
   (b) Lecture #2 Test Activities
      i. List the activities conducted by test engineers
      ii. Draw the software V Model and explain relationships between testing activities and design activities.
      iii. Explain the relationship between acceptance testing, system testing, integration testing, module testing, and unit testing.
      iv. Explain Beizers Testing Levels and Test Process Maturity
      v. Define the relationship between testing and debugging.
      vi. Explain the relationship between the cost of fixing a defect and the phase in which the defect is discovered.
   (c) Lecture #3 Validation and Verification
      i. Define validation and verification.
      ii. Explain the term IV & V.
      iii. Define the terms fault, error, and failure.
      iv. Define testing, test failure, and debugging.
      v. Explain the requirements for a fault to be observed (Reachability, Infection, Propagation)
      vi. Explain the concept of test case values
      vii. Explain expected results
      viii. Explain test set.
      ix. Explain test script.
2. Week #2
   (a) Lecture #1 Testing Termonology
      i. Explain the difference between white box and black box testing.
      ii. Explain the difference between static testing and dynamic testing.
      iii. Define test requirement.
      iv. Define coverage criterion.
      v. Define coverage and coverage level.
      vi. Define infeasibility.
      vii. Compare and contrast white and black box testing.
      viii. Compare and contrast top down and bottom up testing.
      ix. Compare and contrast static testing with dynamic testing.
   (b) Lecture #2 Graph Theory
      i. Define the term graph, node, initial node, final node, and edge.
      ii. Define reachability in a graph.
      iii. Define the term test path.
      iv. Explain the concept of a SESE graph.

v. Relate the concept of a SESE graph to good programming concepts.
vi. Define the concept of visiting a node and explain the concept of a graph tour.
vii. Given a graph, construct a set of test paths through the graph
viii. Construct a graph from a segment of source code.

(c) Lecture #3 Coverage
i. Define node coverage
ii. Define edge coverage
iii. Define edge-pair coverage
iv. Define prime path
v. Define prime path coverage
vi. Construct a set of test paths which meet the criteria for node coverage
vii. Construct a set of test paths which meet the criteria for edge coverage
viii. Practice translating source code into a control flow graph.
ix. Compare and contrast the number of test cases necessary meet the associated testing criteria.

3. Week #3

(a) Lecture #1 Active Learning with Coverage
i. Practice translating source code into a control flow graph.
ii. Compare and contrast the number of test acses necessary to meet the associated testing criteria.

(b) Lecture #2 Testing from Use Cases
i. Explain the purpose of the use case diagram.
ii. Given a use case scenario, construct an activity diagram showing the flow through the use case.
iii. Given an activity diagram, construct a set of test cases to fully exercise the use case.

(c) Lecture #3 Equivalence Class Testing
i. Explain the concept of design by contract and test by contract.
ii. Define equivalence class.
iii. Compare and contrast defensive testing and testing by contract.
iv. Given a software description, define the equivalence classes for a given problem.
v. Given a software description, construct test cases using the equivalence partitioning method.
vi. Based on Equivalence class testing, determine the minimum number of test cases necessary to test a given software system.

4. Week #4

(a) Lecture #1 Boundary Value Analysis.
i. Define a Software Boundary condition
ii. Explain why boundary value 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 system
vi. Construct test cases combining equivalence classes and boundary value testing to test multi-variable problems.

(b) Lecture #2 Introduction to TestNG
i. Explain the difference between JUnit and TestNG
ii. Explain the concept of a data provider.
iii. Construct a set of unit tests using TestNG and data providers.
iv. Compare and contrast the differences in philosophy between JUnit and TestNG.
v. Explain how to use folders to organize test code separate from source code.

(c) Lecture #3 Input Domain testing
i. Explain the concept of an input domain
ii. List the steps necessary to perform input domain modeling.
iii. Compare and contrast interface based input domain modeling with functionality based input domain modeling.

5. Week #5

(a) Lecture #1 Class Cancelled
(b) Lecture #2 Midterm Exam Review
(c) Lecture #3 Midterm Exam