CSCI4490, Software Engineering
Exam II, Fall 2015
Study Guide

- Exam II will be held during class on **Friday, November 6, 2015.**

- Exam II covers Chapters 6, 9, 13, 14, and 15 from the Schach text, plus any handouts provided during class.

- Per the course policy, use of calculators or other electronic devices during the exam is prohibited.

- **Crib Sheet:** The exam is closed book/notes, with the exception that students MAY bring in a **crib sheet** that meets the following specifications:

Exam II Crib Sheet Specifications:

- A single 8.5" x 11" piece of paper may be used (you MAY write on one side ONLY) for the exam.
- The crib sheet must be **handwritten** solely by the person using the crib sheet on the exam, and signed by that person in the top right corner indicating that the crib sheet meets the specifications given here.
- The crib sheet may contain ANY material the author sees fit to include. You are encouraged to critique, discuss and otherwise exchange ideas with other students about what kinds of material would be useful to have on your crib sheet.
- No mechanical reproductions or reductions are allowed.
- Any crib sheet not meeting the above specifications will be confiscated during the exam, with the instructor reserving the right to alter your exam grade as deemed appropriate.

The following exam objectives are intended to aid you in preparation for the course's examinations. However, they are not considered to be inclusive of all the testable material from this course. All assigned reading material, supplemental handouts, class lectures, class discussions, homework, programming projects, team assignments, and in-class problem-solving sessions are considered testable material.

================ Exam II Objectives ================

Non-Execution-Based Testing- Chapter 6 sections 6.1 & 6.2

- Understand the difference between non-execution-based testing and execution-based testing.
- Understand the basic concept of a walkthrough.
- Know the basic composition of a walkthrough team.
- Understand why corrections aren't made during a walkthrough.
- Understand the basic concept of an inspection.

- Know the basic composition of an inspection team.
- Understand the differences between a walkthrough and an inspection.
- Understand the differences between verification and validation and how the products of each workflow can be verified.
- Understand when product validation is conducted. Know the different metrics used during walkthroughs and inspections to assess product and process effectiveness and efficiency.
- Know which metrics can be used to provide an assessment of the product, which can provide an assessment of the process, and which can provide an assessment of both.

Project Management- Chapter 9 plus Handout

- Understand the activities of the Project Management Life Cycle.
- Understand the methods for estimating the size and cost of a software product.
- Understand how to determine the Critical Path from a project schedule.
- Understand the concept of slack time as pertains to a project schedule.
- Understand the two techniques used for resource leveling in order to correct resource over-allocations.

Object Oriented Analysis- Chapter 13

- Understand the orientation (primarily data or primarily action) of the various object-oriented analysis techniques. Understand the three basic steps used in Object-Oriented Analysis (OOA) and what type of activity is performed in each step.
- Be able to utilize the UML notation for inheritance, aggregation, and association as part of the OOA of a software requirement.
- Be able to apply the OOA techniques of use-case modeling, class modeling, and dynamic modeling to a problem solution.
- Understand the relationship between use-cases and scenarios.
- Be able to describe both normal and abnormal scenarios.
- Describe the purpose of a scenario walkthrough.
- Understand when scenario walkthroughs should be stopped.
- Understand the context in which noun extraction would be useful as an OOA technique.
- Be able to apply the noun extraction technique to a given software requirement.
- Understand how nouns that lie outside the problem boundary should be dealt with in the final stage of noun extraction.
- Understand how abstract nouns should be dealt with in the final stage of noun extraction.
- Understand the importance of the concept of "state" regarding the decision of whether to make a noun a class.
- Be able to evaluate the correctness of a given noun extraction for a software requirement.
- Understand the context in which CRC cards would be useful as an OOA technique.
- Discuss the difference between “uses” and “used by” in terms of CRC card collaboration.
- Be able to apply the CRC card technique to a given software requirement.
- Be able to evaluate the correctness of a CRC card class modeling application.
- Know the three types of classes used in class modeling and be able to identify the type of a class given a description of the class.

- Be able to utilize the UML state diagram notation as part of the OOA of a software requirement.
- Be able to evaluate the correctness of a UML state diagram.
- Understand why/how UML guards are used in state diagrams.
- Understand the role of testing during the OOA phase.
- Understand the role of iteration within OOA.

Object-Oriented Design- Chapter 14

- Understand the difference between operation, data, and object-oriented design
- Know the two methods for representing the detailed design of a product and the advantages of each
- Be able to describe the two steps of OOD as presented by the author
- Understand the role of scenarios within OOD and their relationships to interaction diagrams constructed as part of determining the methods in the detailed class diagram
- Understand the two types of interaction diagrams used in object-oriented design and the difference between them
- Be able to use the UML sequence diagram notation as part of the OOD of a software requirement
- Be able to evaluate the correctness of a UML sequence diagram
- Be able to use the UML collaboration diagram notation as part of the OOD of a software requirement
- Be able to evaluate the correctness of a UML collaboration diagram
- Be able to convert between UML collaboration and sequence diagrams when representing a scenario during OOD
- Be able to produce a UML sequence or collaboration diagram to complete the details of a class diagram
- Understand the things to consider when assigning a method to a class
- Be able to use the UML detailed class diagram notation as part of the OOD of a software requirement
- Be able to evaluate the correctness of a UML detailed class diagram
- Be able to create the detailed design for a class’s methods from a statechart diagram of the class
- Understand the goal of testing during the design workflow and how to show whether the Design correctly reflects the Specification
- Understand the metrics used during OOD
- Understand the difficulties associated with the design of real-time systems

Implementation- Chapter 15 sections 15.1 through 15.5 and 15.7

- Understand the factors that must be taken into consideration when choosing a programming language for implementation.
- What is the overriding thing to consider when choosing a programming language?
- Know the difference between 1st, 2nd, 3rd, and 4th generation languages.
- Understand the potential benefits and disadvantages/dangers of using a 4th generation language for implementation.

- Be able to identify several tenets of good programming practice.
- Understand the advantages and disadvantages of using programming standards.

Integration- Chapter 15 section 15.6

- Be able to discuss the various approaches to program integration:
  - Implementation then Integration
  - Top-down Integration
  - Bottom-up Integration
  - Sandwich Integration
- Be able to discuss the advantages and disadvantages of the each of the above approaches to program integration.
- Be able to discuss the approach to program integration most appropriate for the integration of object-oriented products.
- Who is best suited to provide oversight of the integration process? Why?