IC470, Software Engineering

Exam: In class as per the course syllabus

- As per the course policy, only non-programmable calculators may be used on the exam.
- The exam will cover chapters 1, 2, and 13 of the Schach text as well as material covered in the milestones and labs.
- The following exam objectives are intended to aid you in your preparations for the course's examinations, but 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, labs, team assignments, and in-class problem-solving sessions are considered testable material.

================= 6 Week Exam Objectives =================

Basic software development objectives

- Identify the phases of the software life cycle.
- Describe the activities inherent in each life cycle phase.
- Discuss the relative costs of each life cycle phase.
- Understand the relative cost savings associated with uncovering faults in these various life-cycle phases.
- Apply knowledge of the costs of the software life cycle for planning a software development project.

Software Life Cycles/Requirements

- Compare and contrast software development life cycle models such as Build-and-Fix, Waterfall, Spiral, and Agile Software Development. For each model understand if/how the phases identified in the 1968 NATO conference that resulted in the Waterfall Model are used in the respective models.
- Understand the strengths and weaknesses of the various software life cycle models.
- Describe the role of risk analysis within the spiral model of software development.
- Given system requirements, be able to develop a functional requirements trace table, to include both functional requirements as well as acceptance test plan test case scenarios that can be used to objectively validate that the functional requirements have been met once the system has been implemented.
- Be able to distinguish between “normal” “abnormal, and not useful test cases.
- Develop a project planning Gantt Chart for a software development project.

Object Oriented Analysis

- Understand Use Case modeling and Class modeling as part of OOA.
- Apply the OOA techniques of use-case modeling and class modeling using correct UML notation.

- Understand the role of scenario diagrams within OOA.
- 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 a OOA technique
- Apply the noun extraction technique to given software requirements
- 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 significance of the concept of "state" regarding the decision of whether to make a noun a class
- Discuss the abstraction levels on which the various noun extraction stages focus.
- Evaluate the correctness of a given noun extraction for a software requirement
- Understand the principles of information hiding in which the information stored in a database is abstractly represented as separate from the database and the rest of the system.
- Describe the role of testing during the OOA phase
- Describe the role of iteration within OOA
- Apply Data Flow Diagram techniques to non-objected-oriented modeling situations, such as complex hardware interactions.

Project Plan

- Be able to read a Gantt chart
- Understand the project planning information communicated by a Gantt chart
- Describe the use of a Gantt chart as a project planning tool, both for planning and for monitoring actual progress during project development
- Be able to construct a Gantt chart based on estimates of the time required to complete the various tasks of a project and the software development method being used
- Understand the impact on project planning and Gantt chart construction of various software lifecycles such as Build-and-Fix, Waterfall, Spiral Model, and Agile development

Related papers given in the milestones (see milestones for electronic versions of these papers).

- Describe the ramifications to software development discussed in Brook's "No Silver Bullet" article.
- Understand the ramifications to software development resulting from the Therac-25 incidents discussed in Leveson’s "Medical Devices: The Therac-25" article.
- Discuss the ramifications to software development presented by the emergence of Agile Software Development as given in Fowler’s “The Agile Manifesto,” Ferraria’s “Agile Development Iterations and UI Design,” and Paton’s “Hitting the Target: Adding Interaction Design to Agile Development”

- Apply risk analysis techniques as described in Boehm’s “A Spiral Model of Software Development and Enhancement”