CSCI 5313 - SOFTWARE DEVELOPMENT PRINCIPLES

CREDIT HOURS: 3
PREREQUISITES: Nine advanced hours of CSCI
GRADE REMINDER: Must have a grade of C or better in each prerequisite course.

CATALOG DESCRIPTION

State-of-the-art principles of software design and development. Theories, methodologies, techniques, and tools of software engineering. Case studies.

PURPOSE OF COURSE

To provide the student with a knowledge of state-of-the-art software design and development principles with a software engineering orientation.

EDUCATIONAL OBJECTIVES

Upon successful completion of the course, students should be able to:

1. Explain the need for software engineering principles and techniques in software development and maintenance.

2. Independently perform research on an advanced topic in software engineering and report on it orally and in writing.

3. Describe the major activities of software development.

4. Choose an appropriate software process model for a particular development effort.

5. Use modern techniques of requirements analysis, design, and implementation.

6. Appreciate the historical evolution of software engineering techniques.

COURSE CALENDAR

This course meets for a minimum of 37.5 lecture contact hours during the semester. Students have significant weekly extracurricular assignments which may involve reading, teamwork and team meetings, or engaging in other forms of preparation. Students are expected to complete homework assignments, programming assignments, projects and presentations, and 2-3 periodic exams in addition to the final exam. Students are expected to prepare for any class assignments or quizzes over the material covered in class or the extracurricular assignments. Successful completion of these activities requires at a minimum six additional hours of outside of classroom work each week.

CONTENT **Hours**

Introduction.....3
- History of software design and development
- Software crisis
- Software life cycle

Management issues
Origins of Software Engineering

Requirements Engineering 5

Design Methods 12

Software Process, Software Metrics, Tools 9

Implementation and Testing 10
Programming environments, teams, languages, and style
Strategies for maintainability and reusability
Test case design, program testing, and system testing
Quality assurance, verification, validation, reliability

Evolution.....3
Operation; performance analysis and measurement
Maintenance

Exams (plus final) 3

TOTAL 45

REFERENCES

Pressman, R. S., Software Engineering: A Practitioner's Approach, 6th Ed., McGraw-Hill, 2005.

Schach, S. R., Object-Oriented and Classical Software Engineering, 7th Ed., McGraw-Hill, 2007.

Sommerville, I., Software Engineering, 8th Ed., Addison-Wesley, 2007.