COMPUTER SCIENCE 5704
SOFTWARE ENGINEERING
(ADP TITLE: SOFTWARE ENGINEERING)

I. CATALOG DESCRIPTION:

5704 SOFTWARE ENGINEERING

Study of the principles and tools applicable to the methodical construction and controlled evolution of complex software systems. All phases of the life cycle are presented; particular attention focuses on the design, testing, and maintenance phases. Introduction to software project management. Attention to measurement models of the software process and product which allow quantitative assessment of cost, reliability, and complexity of software systems.

(3H,3C).

II. LEARNING OBJECTIVES:

Having successfully completed this course, the student will be able to apply contemporary techniques and tools to the design and testing of large-scale software systems and assume the role of a productive member on a multi-person development team. The student will also be able to apply contemporary models to predict development costs, measure and predict reliability, and measure software complexity.

III. JUSTIFICATION:

The development and maintenance of software systems is, and will continue to be, an important single item in our national economy and defense establishment as well as a key component in our international trade. Leadership in high technology areas is closely associated with sustained increases in our ability to overcome the current limitations on the production of high quality software systems with predictable and reasonable costs. The term "software engineering" was coined in 1979 to generally describe the concepts, techniques and tools relating to this goal. Many graduate students aim at careers in industry where they work as software developers and, later in their careers, as managers of software projects. It is critical for such students to develop a sound understanding of how to use contemporary methods that are part of the software engineering process. Students in other sub-areas of applied computer science will also benefit from this course because progress in these other sub-areas in many cases involves the construction of software systems.

Software engineering has evolved into a mature subdiscipline within computer science, with a solid foundation of principles and tools for all phases of the software life cycle. Many of these principles for design and analysis of software systems are introduced in 5704. Thus, the prerequisite 5034 has been removed. The required texts have been updated along with the inclusion of requirements analysis in the syllabus.

IV. PREREQUISITES AND COREQUISITES:

Materials covered in this course presume a fundamental background in undergraduate-level computer science and programming. The catalog statement on prerequisites associated with course level is thus appropriate.

V. TEXTS AND SPECIAL TEACHING AIDS:

Required text to be chosen from:

Blum, Bruce I. SOFTWARE ENGINEERING: A HOLISTIC VIEW. New York, NY: Oxford, 1992. xiii, 588.

Brooks, Fred. MYTHICAL MAN MONTH. Reading, Massachusetts: Addison-Wesley Publishing Company, 1982. xi, 195.

Pressman, Roger S. SOFTWARE ENGINEERING: A PRACTIONER'S APPROACH, 4th Ed. New York, NY: McGraw-Hill, 1997. xxvii, 852.

VI. SYLLABUS:

	Percent of Course
1. Life cycle models	5%
2. Requirements Analysis	5%
3. Software design methods a. control-based design b. data oriented design c. other methods (information hiding, dialogue oriented) d. object oriented design	30%
4. Testing a. testing theory b. measures of degree of testing c. test case development techniques d. testing tools	30%
5. Configuration Management	5%
6. Role of models and measures in software engineering a. complexity models b. code level models/measures c. structure level models/measures d. case studies e. cost and reliability	20%
7. Project Management a. data collection/monitoring b. project organization and feedback	5%
	100%

VII. OLD (CURRENT) SYLLABUS:

	Percent of Course
1. Life cycle models	5%
2. Software design methods a. control-based design b. data oriented design c. other methods (information hiding, dialogue oriented) d. object oriented design	35%
3. Testing a. testing theory b. measures of degree of testing c. test case development techniques d. testing tools	30%
4. Configuration Management	5%
5. Role of models and measures in software engineering a. complexity models b. code level models/measures c. structure level models/measures d. case studies e. cost and reliability	20%
6. Project Management a. data collection/monitoring b. project organization and feedback	5%
	100%

VIII. CORE CURRICULUM GUIDELINES:

NA