CSC 323 - SOFTWARE ENGINEERING

CREDIT HOURS: 3
PREREQUISITES: CSC 241
GRADE REMINDER: Must have a grade of C or better in each prerequisite course.

CATALOG DESCRIPTION


PURPOSE OF COURSE

To provide the student with a knowledge of software engineering principles that can be applied to the software process.

EDUCATIONAL OBJECTIVES:

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

1. Identify software development problems that provided the impetus for the start of software engineering.
2. Demonstrate an understanding of the different perspectives from which software is considered by users, clients, and commercial and in-house developers.
3. Describe the importance of software maintenance, and the nature of the software life cycle.
4. Describe the various software process models that have been used for software development and gain familiarity with important software development methodologies.
5. Work in a disciplined software development team demonstrating the use of COCOMO, function points, and other methods to estimate the size of a development effort.
6. Produce important artifacts of software development other than code.
7. Demonstrate an understanding of the role of software quality assurance and practice non-execution based testing.
8. Develop a prototype as a means of requirements validation.
9. Derive and use metrics for software development.
10. Use state-of-the-practice software estimation techniques.

CONTENT

Introduction ................................................................. 3
   History of software engineering
   The need for a disciplined approach
   Software process models

Software Engineering Issues .................................................... 3
   Quality, productivity, accuracy, reliability, maintainability, reusability
   The use of metrics
   The role of Computer-Assisted Software Engineering (CASE)

Requirements Engineering .................................................... 10
   Requirements definition and analysis
Feasibility study
Cost/benefits analysis
Prototyping
Tools

Design .................................................. 12
Methodologies: structured design, functional decomposition, data-flow oriented, data-oriented, object-oriented design
Tools

Implementation and Testing ................................. 10
Programming environments, teams, languages, and style
Programming principles: cohesion, coupling, modularity, information hiding
Test case design, classes of tests
Quality assurance, verification, validation, reliability
Testing methods
Tools

Evolution .................................................. 4
Operation; performance analysis and measurement
Maintenance
Reverse engineering

Exams (plus final) ........................................... 3

TOTAL 45

REFERENCES
