CS A220: SOFTWARE ENGINEERING

Item	Value
Curriculum Committee Approval Date	12/06/2023
Top Code	070600 - Computer Science (Transfer)
Units	4 Total Units
Hours	108 Total Hours (Lecture Hours 54; Lab Hours 54)
Total Outside of Class Hours	0
Course Credit Status	Credit: Degree Applicable (D)
Material Fee	No
Basic Skills	Not Basic Skills (N)
Repeatable	No
Grading Policy	Standard Letter (S), • Pass/No Pass (B)

Course Description

Introduction to the concepts, methods, and current practice of software engineering. Study the lifecycle of a software system. Employ engineering methods, processes, techniques, and measurement. Use of tools to manage software development. Project work is required to illustrate the various elements. PREREQUISITE: CS A150; and CS A100, CS A122, CS A131 or CS A170. Transfer Credit: CSU; UC.

Course Level Student Learning Outcome(s)

1. Apply the system development life cycle steps in designing systems from module design through its implementation.
2. Evaluate existing systems using the software engineering process.

Course Objectives

- 1. List the different life cycles and their appropriateness in different situations, define the basic principles of software engineering and recall how they apply throughout the software life cycle.
- 2. Report the tradeoffs and relationships among the various activities in the software life cycle and restate the meaning and use of a set of basic software qualities.
- 3. Arrange interviews with customer to elicit requirements, formulate the requirement document, and describe its structure and the appropriate kinds of information in such a document.
- 4. Record the differences among interaction patterns of a set of basic architectural styles, between architecture and module design, and employ an appropriate architectural style for a particular problem.
- 5. Use provided/exported and required/imported interfaces to define module boundaries, identify and define modules, abstract data types, coupling, cohesion, fan-in, and fan-out in a design.
- 6. Construct a design for a nontrivial, sizable problem.
- 7. Set up a module design onto an implementation in source code, employ existing modules and libraries in an implementation, and construct code under a heavy deadline.
- 8. Evaluate a program for failures, apply white-box testing or black-box testing on short pieces of code, recognize the many dimensions of software quality assurance, inspection, and code walk-through process.

Lecture Content

Overview Introduction FAQs about software engineering Professional and ethical responsibility Computer-Base System Engineering Systems and environment System modeling The system engineering process Software Process Software process models Process Iteration Software specification Software design and implementation Software validation and evolution Automated process support Project Management Project planning and scheduling Risk management Requirements Software requirements Functional and non-functional requirements User requirements System requirements The software requirements document Requirements engineering process Feasibility study Requirements elicitation and analysis Requirement validation and management System models Context, behavioral, Data, and Object model CASE workbenches Design Architectural Design System structuring Control models Modular decomposition Domain-specific architecture Distributed Systems Architectures Multiprocessor, client-server, and distributed object architectures COBRA Objective-Oriented Design Objectives and classes Object-oriented design process Design evolution Design with reuse Component-based development Application families Design Patterns User interface design User interface design principles User interaction Information pres entation User support Interface evolution Verification and Validation Verification and validation Verification and validation planning Software inspection Software testing Defect testing Integration testing Object-oriented testing Testing workbenches Evolution Software change Program evolution dynamics Software maintenance Architectural evolution Configuration management Configuration management planning Change management Versions and release management System building CASE tools for configuration management

Method(s) of Instruction

- Lecture (02)
- DE Live Online Lecture (02S)
- DE Online Lecture (02X)
- Lab (04)
- DE Live Online Lab (04S)
- DE Online Lab (04X)

Instructional Techniques

Lecture Problem solving PowerPoint presentations Discussion

Reading Assignments

Students will spend a minimum of 4 hours per week reading the textbook and/or other reading material assigned.

Writing Assignments

Students will spend a minimum of 6 hours weekly completing programming assignments and project presentations.

Out-of-class Assignments

Students will spend a minimum of 6 hours per week completing weekly programming assignments.

Demonstration of Critical Thinking

Tests and quizzes Homework assignments In-class assignments Software projects

Required Writing, Problem Solving, Skills Demonstration

Successful performance of the assignments and project presentations.

Eligible Disciplines

Computer science: Masters degree in computer science or computer engineering OR bachelors degree in either of the above AND masters degree in mathematics, cybernetics, business administration, accounting or engineering OR bachelors degree in engineering AND masters degree in cybernetics, engineering mathematics, or business administration OR bachelors degree in mathematics AND masters degree in cybernetics, engineering mathematics, or business administration OR bachelors degree in any of the above AND a masters degree in information science, computer information systems, or information systems OR the equivalent. Note: Courses in the use of computer programs for application to a particular discipline may be classified, for the minimum qualification purposes, under the discipline of the application. Masters degree required.

Textbooks Resources

1. Required Tsui, Frank. Essentials of Software Engineering, 5th ed. Marietta: Jones Bartlett, 2022