SOFTWARE ENGINEERING (SOFT)

SOFT 423 Software Requirements Units: 3.00
An integrated approach to discovering and documenting software requirements. Identification of stakeholders; customer, operator, analyst, and developer perspectives. Requirements elicitation. Transition from initial (informal) requirements to semi-formal and formal representations. Requirements analysis process; analysis patterns. Requirements specification techniques. Relation to architecture and user interface design; traceability of requirements. Alternately offered as CISC 423.
(Lec: 3, Lab: 0, Tut: 0)
Requirements: Prerequisites: CMPE 223 Corequisites: CMPE 322 Exclusions: CISC 423
Offering Term: W
CEAB Units:
Mathematics 0
Natural Sciences 0
Complementary Studies 0
Engineering Science 24
Engineering Design 12
Offering Faculty: Smith Engineering
Course Learning Outcomes:

1. Understand the process of requirements development, including elicitation, analysis, specification and validation.
2. Understand and identify different types of software requirements.
3. Elicit requirements information about the system from a variety of sources using a variety of techniques.
4. Use requirement analysis and modeling techniques to analyze software requirements.
5. Write software requirement specification documents and prepare validation plans to validate the final product.
6. Use change control and requirement tracing techniques to minimize disruptive impact of requirement changes.
7. Have the capability to perform requirements elicitation, analysis, specification, and validation for software projects.

SOFT 437 Performance Analysis Units: 3.00
Analytic and empirical evaluation of the performance of software systems. Performance modeling. Experimental design and statistical techniques for empirical performance analysis. Alternately offered as CISC 437.
(Lec: 3, Lab: 0, Tut: 0)
Requirements: Prerequisites: CMPE 324 (CISC 324) or ELEC 377, or permission of the instructor Corequisites: Exclusions:
Offering Term: F
CEAB Units:
Mathematics 0
Natural Sciences 0
Complementary Studies 0
Engineering Science 24
Engineering Design 12
Offering Faculty: Smith Engineering
Course Learning Outcomes:

1. Analyze software architecture and design to identify performance problems.
2. Apply performance oriented principles, performance patterns and anti-patterns in designing real life software systems.
3. Design software systems to meet performance criteria.
4. Learn the performance issues that arise in the real-world, large-scale software system.
5. Understand performance oriented principles, performance patterns and anti-patterns.
6. Understand the basic concepts on designing high performance software systems, data collection techniques, software measurement and instrumentation techniques.
7. Create models to estimate software performance in the architecture and design levels.