Miami Dade College

Course Description

CEN4025C | Software Engineering II | 4.00 credits

This upper division course is for students majoring in the B.S. in Information Systems Technology or the B.S. in Electrical and Computer Engineering Technology programs. This course covers in-depth topics in software process structures, process models, requirements modeling with use-cases and class-based methods. Students will also learn design concepts including abstraction, OOD concepts, component-level and architectural design, user interface analysis and design, and design patterns. Prerequisite(s): CET3383C.

Course Competencies:

Competency 1: The student will demonstrate knowledge of the software process by:

1. Describing the differences between the prescriptive process models, including waterfall, incremental, evolutionary, and others
2. Describing the principles behind agile development methods, extreme programming and agile unified process
3. Explaining the human aspects of software engineering, including the makeup of the SW team and team structures

Competency 2: The student will demonstrate an understanding of and proficiency in software modeling and requirements-gathering activities by:

1. Describing the core principles that guide each framework activity, including planning, modeling, construction, and deployment
2. Eliciting requirements by identifying stakeholders and developing use cases
3. Building the analysis model, creating analysis patterns, and validating requirements
4. Performing domain analysis and applying requirements modeling approaches, including scenario- and class-based methods
5. Writing use cases
6. Developing and writing the software requirements specification document

Competency 3: The student will demonstrate an understanding of software design activities and OOA/D by:

1. Applying design concepts, including abstraction, design patterns, information-hiding principles, refactoring, and designing for testing
2. Creating a domain model, identifying and listing conceptual classes
3. Writing a formal Software Design Specification document
4. Developing class diagrams
5. Identifying and listing associations in the domain model
6. Creating system sequence diagrams and operational contracts
7. Refining the architecture of the system into components
8. Designing with layers and applying the model-view separation principle to develop software architectures

Competency 4: The student will demonstrate an understanding of pattern-based design by:

1. Conducting component-level design, including design and functional design at the component level using design patterns
2. Describing the fundamental design patterns and when they can be applied
3. Develop responsibility-driven designs by creating objects with responsibilities

Competency 5: The student will demonstrate an understanding of object design with GRASP by:

1. Describing what is designed with GRASP

2. Describing the connection between responsibilities, GRASP and UML diagrams
3. Applying GRASP to object design
4. Applying the GRASP patterns: Creator, Information Expert, Low Coupling, Controller, and High Cohesion to a small application

Competency 6: The student will demonstrate how to map their designs to code by:

1. Creating class definitions from design class diagrams
2. Creating methods from interaction diagrams
3. Writing exception handlers for dealing with run-time exceptions
4. Explaining test-driven or test-first development

Competency 7: The student will demonstrate an understanding of performing architectural analysis and refinement of the logical architecture of the system by:

1. Identifying and analyzing the non-functional requirements/architectural factors that impact the architecture
2. Developing quality scenarios that define measurable/observable responses that can be verified (i.e.,) developing quality scenarios of the form
3. Analyze alternatives and create solutions that resolve the impact
4. Designing for separating concerns to maximize low coupling and high cohesion at the architectural level
5. Applying Façade, Observer, and Controller patterns in architectural layers
6. Organizing packages to reduce the impact of changes to the system

Learning outcomes:

- Use quantitative analytical skills to evaluate and process numerical data
- Solve problems using critical and creative thinking and scientific reasoning
- Use computer and emerging technologies effectively