Miami Dade College

Course Description

CEN4065C | Software Architecture and Design | 4.00 credits

This upper division course is for students majoring in B.S. in Information Systems Technology or B.S. in Electrical and Computer Engineering Technology. The course covers the software engineering process of building the software architecture and designing the software product according to design criteria. This course covers in-depth topics in software process models, software architecture, requirements modeling with use-cases and class-based methods. Students will also learn design concepts including abstraction, component-level and OO architectural design, user interface analysis and design, and design patterns, and software project management. Prerequisite: COP 3530.

Course competencies:

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

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

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

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

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

- Applying design concepts including abstraction, design patterns, information-hiding principles, refactoring, and designing for test.
- Writing a formal software design specification document.
- Using the UML as a language to:
    - Describe a domain model
    - Identifying and listing conceptual classes
    - Develop class diagrams
    - Identify and list associations in the domain model ○ Create system sequence diagrams and operational contracts
    - Create interaction and activity diagrams to model the dynamic states of the system
- Refining the architecture of the system into components
- 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:

- Conducting component-level design, including design and functional design at the component level using reusable patterns that describe solutions to recurring problems in software engineering • Describing the basic and reusable design patterns and when they can be applied

- Developing responsibility-driven designs by creating objects with responsibilities

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

- Describing what is designed with a GRASP
- Describing the connection between responsibilities, GRASP, and UML diagrams
- Applying GRASP to object design
- 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:

- Creating class definitions from design class diagrams
- Creating methods from interaction diagrams
- Writing exception handlers for dealing with run-time exceptions • 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:

- Identifying and analyzing the non-functional requirements/architectural factors that have an impact on the architecture
- Developing quality scenarios that define measurable/observable responses that can be verified (i.e., developing quality scenarios of the form
- Analyzing alternatives and creating solutions that resolve the impact
- Designing for a separation of concerns to maximize low coupling and high cohesion at the architectural level
- Applying the use of façade, observer, and controller patterns in the context of architectural layers
- Organizing packages to reduce the impact of changes to the system

Competency 8: The student will demonstrate an understanding of software project management and processes by:

- Creating the work breakdown structure (WBS), the project scheduling, and developing a risk assessment and management plan.
- Applying software estimation techniques, including:
    - Function point analysis ○ Use case points
    - Story points in agile

- Performing risk management activities, including:
    - Risk identification
    - Assessment
    - Mitigation.

- Using software effort estimation tools to determine the level and the scope of a software project in terms of man-months and lines of code, and using project management software (e.g., Microsoft Project, Jira, etc.) to manage the project.