Miami Dade College

Course Description

CET3383C | Software Engineering I | 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 student will learn the basic principles and concepts of software engineering; system requirements; modeling and testing; object-oriented analysis and design; testing and validation; configuration management; and the analysis, design and programming of extensible software systems. Prerequisite: CET2369C or COP2800.

Course Competencies:

Competency 1: The student will demonstrate knowledge of software requirements activities by:

1. Using appropriate methods to elicit requirements from stakeholders
2. Performing analysis of requirements for feasibility of development
3. Checking requirements for accuracy, lack of ambiguity, completeness, consistency, and traceability
4. Constructing and analyzing prototypes
5. Using appropriate methods for management of requirements, including configuration management

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

1. Determining the process and strategy to be used in software design (such as top-down or bottom-up, stepwise refinement, use of patterns and pattern languages, iterative and incremental processes)
2. Selecting and applying the appropriate design methodology (such as a structural or object-oriented approach)
3. Considering and weighing design alternatives and performing trade-off analysis
4. Managing software design activities and requirements change

Competency 3: The student will demonstrate an understanding of software construction activities by:

1. Selecting appropriate processes and models for constructing software, including appropriate reuse processes
2. Selecting appropriate frameworks, platforms, and environments
3. Establishing and following project standards for version control and configuration management
4. Creating detailed designs that minimize complexity and enhance quality
5. Writing code to implement detailed designs, and refactoring the code when needed
6. Using defensive coding techniques to minimize propagation of errors and threats
7. Using appropriate design patterns
8. Documenting code through comments to support software maintenance

Competency 4: The student will demonstrate an understanding of software testing activities by:

1. Identifying all stakeholders involved in software testing
2. Designing and implementing the software test plan
3. Identifying tools to be used throughout testing activities
4. Designing/selecting and implementing the test environment
5. Designing, implementing, and executing test cases
6. Identifying test objectives
7. Identifying, collecting, and storing appropriate data resulting from testing/demonstration
8. Identifying, assigning, and performing necessary corrective actions
9. Analyzing test data for test coverage, test effectiveness, and process improvement

Competency 5: The student will demonstrate an understanding of software process life cycle activities by:

1. Describing organization-wide life cycle models (such as waterfall, spiral, V-model, incremental, capability maturity models, etc.)
2. Describing process activities that take place within a specified life cycle process model
3. Designing or tailoring a software process to the needs of a project team or software engineering activity.
4. Implementing and executing software processes
5. Defining guidelines for software teams on how to implement and manage software processes.
6. Collecting data for assessment of a software process
7. Using assessment information and reports for software process improvement

Competency 6: The student will demonstrate an understanding of software quality activities by:

1. Identifying, establishing, following, and verifying appropriate processes, standards, and quality models that facilitate achieving quality goals and attributes
2. Identifying and using appropriate tools and measurements needed to reach quality goals and attributes.
3. Establishing and executing corrective actions if quality goals are not achieved
4. Establishing and executing appropriate continuous improvement processes
5. Establishing and updating requirement traceability metrics and verification metrics

Competency 7: The student will demonstrate an understanding of software configuration management (SCM) by:

1. Determining the organizational context for and constraints imposed on SCM
2. Identifying software components to be controlled by SCM
3. Designing data and code repositories
4. Planning versioning procedures for path branching and path integration
5. Developing/adopting a change control process
6. Generating, classifying, and managing problem/defect reports
7. Identifying and developing a criteria for selecting SCM tools
8. Developing the process for establishing a SCM library
9. Developing a software release plan

<u>Learning Outcomes:</u>

- Solve problems using critical and creative thinking and scientific reasoning
- Use computer and emerging technologies effectively

Updated; Fall 2025