CSC 301 Software Engineering II  

Catalog description:
This course is an extension of CSC 300 and focuses on the implementation of the software engineering principles covered therein. It will explore state-of-practice and cutting-edge techniques and tools related to the design, implementation and maintenance of software systems. Topics include: design patterns; Model Driven Architecture (MDA); test-driven development; agile development; extreme programming (XP); aspect-oriented design. An ongoing group project will be used to gain practical experience with current software engineering practices and a variety of IDEs and CASE tools. Three lecture hours per week and three hours of scheduled laboratory per week, plus programming work outside of class.

Prerequisite: CSC 300; CSC 263 strongly recommended.

Goals:
The purpose of this course is to develop students’ understanding of modern methodologies, processes and techniques encountered in the development of large-scale software systems. The goals of this course are:

CG1: to give students experience with a variety of software engineering techniques and paradigms;
CG2: to expand and integrate students’ knowledge and skills in the areas of system analysis and software design, implementation and verification;
CG3: to give students experience in making and critiquing presentations;
CG4: to give students experience in team software development.

Upon completion of the course, a student should have experience with a variety of the activities and techniques necessary to conduct the development of a large system, should be able to select and apply the appropriate tools required to effect the development process, and should have an appreciation of the strengths and weaknesses of the various design and implementation models extant in the field.

Objectives:
Upon successful completion of the course, student will have:

CO1: demonstrated understanding of the software development life cycle and its phases;
CO2: demonstrated knowledge of the major techniques and models used in the implementation of each phase (workflow) of software development;
CO3: gained experience with the tools and techniques of software development;
CO4: demonstrated understanding of modern design paradigms;
CO5: properly utilized modern CASE tool environments, specifically including UML modeling, in the design and implementation of a large-scale project;
CO6: participated in the development and presentation of group projects.

<table>
<thead>
<tr>
<th></th>
<th>PO1</th>
<th>PO2</th>
<th>PO3</th>
<th>PO4</th>
<th>PO5</th>
<th>PO6</th>
<th>PO7</th>
<th>PO8</th>
<th>PO9</th>
<th>PO10</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO01</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CO02</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CO03</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CO04</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CO05</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CO06</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Note: Body of Knowledge (BoK) hours indicate the total amount of time the topic is covered in the course: the total for this course is 53. This course has approximately 40 lecture hours of student contact plus approximately 40 laboratory hours of student contact. Some of the 53 hours are covered during scheduled lab, which students are required to attend. Many of the topics covered in this course, particularly those in the SE categories, materially benefit from discussion in a hands-on environment. BoK hours in bold typeface represent core (required) BoK units; BoK hours in regular typeface represent non-core (optional) BoK units.
Topics:

- **Requirements**
  - determining user needs
  - distinguishing "needs" and "wants"
  - overview of the requirements workflow
  - defining scope
  - understanding the domain
  - requirement elicitation techniques
    - interviewing, forms collection, use cases, prototyping
  - test workflow in the context of requirements
  - human factors
  - prototypes and reuse
  - metrics for the requirements workflow
    - what to measure, how to evaluate

- **Object-Oriented (OO) Analysis**
  - overview of the analysis workflow
  - OO analysis overview
  - recognizing entity (data) classes
  - entity modeling
    - noun extraction
    - CRC cards
  - functional modeling
  - dynamic modeling
  - test workflow in the context of OO analysis
  - interface class extraction
  - control class extraction
  - use cases and dynamic modeling
  - the specification document in OO analysis
  - CASE tools for OO analysis
  - metrics for the OO analysis workflow

- **Classical ("Structured") Analysis**
  - structured analysis overview
  - informal specifications
  - the Specification Document
  - structured systems analysis
    - data flow diagrams, alternative techniques
  - entity-relationship (ER) modeling
  - finite state machines
  - other formal techniques, including petri nets, Z, and Anna
  - test workflow in the context of classical analysis
  - the specification document in OO analysis
  - CASE tools for structured analysis
  - metrics for the structured analysis workflow

- **Design**
  - design and abstraction
  - overview of the design workflow
  - operation-oriented (function-oriented) design
  - data flow analysis
  - transaction analysis
  - data-oriented design
  - object-oriented design
• test workflow in the context of design
• real-time design techniques
• CASE tools for design
• metrics for the design workflow

• Design Patterns
  • what is a design pattern?
  • design patterns solve design problems
  • design by contract / programming to an interface
  • design with change in mind
  • toolkits
  • frameworks
  • foundation creational patterns
    • abstract factory, builder, factory method, prototype, singleton
  • foundation structural patterns
    • adapter, bridge, composite, decorator, facade, flyweight, proxy
  • foundation behavioral patterns
    • chain of responsibility, command, interpreter, iterator, mediator, memento, observer, state, strategy, template method, visitor
  • how to select design patterns
    • understand that design patterns are abstractions
    • know each pattern's intent
    • know how patterns interrelate
      → non-trivial problems are likely to require multiple patterns
    • know how similar patterns differ
    • know the causes for redesign (refactoring) and consider patterns designed to avoid those causes
  • what to expect from patterns
  • implementing design patterns
  • CASE tools for design patterns

• Implementation
  • overview of the implementation workflow
  • choosing a programming language / platform
  • good programming practices
    • mnemonic names, self-documenting code, formatting, general style rules
  • coding standards
  • code reuse
    • licensing / intellectual property issues
  • unit integration
  • test workflow in the context of implementation
    • testing to specifications (black box)
    • testing to code (white box, glass box)
    • theory vs. reality of testing
  • black box testing techniques
  • glass box testing techniques
  • unit testing
  • regression testing
  • code walkthroughs and code inspections
  • potential testing problems
  • when to rewrite vs. debug
  • integration testing
  • product testing
  • acceptance testing
  • CASE tools for implementation, testing and code/configuration management
  • metrics for the implementation workflow

• Maintenance, Post-Delivery
The emphasis of the course is on the proper design, management and implementation of a software system from initial conception to final product maintenance. There will be an ongoing case study presented in depth, paralleled by a semester-long project in which all phases of the creation of a moderate-sized system will be addressed by groups within the class. Extensive laboratory work, group discussion time and group presentations conducted as part of the scheduled laboratory sessions are an integral component of the course, serving to reinforce the concepts and techniques presented in lecture.

All programs must conform to departmental guidelines for program design and implementation, and all lab reports must conform to guidelines announced in class. Regardless of numeric average, a student will not be eligible for a passing grade unless he or she has submitted a lab report for every programming assignment.

The course grade will be determined using the following approximate weights: project reports and deliverables: 25%; presentations: 10%; midterm and final exam: 40%; homework and/or papers: 25%.

<table>
<thead>
<tr>
<th>CO01</th>
<th>Exam / Quiz Questions</th>
<th>Homewor k Problems</th>
<th>Programming Projects</th>
<th>Lab Exercises</th>
<th>Group Projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CO02</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CO03</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CO04</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CO05</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CO06</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Web Resources:
- The Institute of Electrical and Electronics Engineers (IEEE). [http://www.ieee.org/portal/site](http://www.ieee.org/portal/site)

Bibliography:

Dikel, David M.; Kane, David; Wilson, James R. **Software Architecture: Organizational Principles and**
Patterns. Prentice-Hall, 2001
Fowler, Martin. Analysis Patterns: Reusable Object Models. Addison-Wesley, 1997
Gamma, Helm, Johnson & Vlissides. Design Patterns. Addison-Wesley, 1995
Kerievsky, Joshua. Refactoring to Patterns. Addison-Wesley Professional, 2004
Larman, Craig. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. Pearson Education, 2005
Shalloway, Alan; Trott, James. Design Patterns Explained: A New Perspective on Object-Oriented Design. Addison-Wesley, 2002