Catalog description:
This course will explore in detail the software development process for large software systems using modern software engineering principles. Topics include software life cycle models, tools and techniques for software engineering, the software development life cycle, the Unified Process, testing/evaluation techniques, and evaluation metrics. Group design projects will be used to gain understanding of course topics and experience with development tools. Three lecture hours and three hours of scheduled laboratory per week, plus programming work outside of class. Not open to students who have received credit for CSC 265 or CSC 266.

Prerequisite: CSC 260.

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 develop an appreciation for the process of large-scale software development;
CG2: to develop the skills and knowledge necessary to analyze, design, verify and document large software systems;
CG3: to give students experience in making and critiquing presentations;
CG4: to give students experience in team software development activities;
CG5: to develop students' writing skills in the context of all aspects of the software engineering process.

Upon completion of the course, a student should know the activities and techniques necessary to conduct the development of a large system, and should be able to select and apply the appropriate tools required to effect the development process.

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

CO1: demonstrated knowledge of the software development life cycle and its aspects and phases;
CO2: demonstrated knowledge of the major models used in the development of large-scale software;
CO3: demonstrated an appreciation of the factors effecting team selection and performance;
CO4: demonstrated knowledge in the tools and techniques of software development, specifically including UML;
CO5: demonstrated knowledge of modern design paradigms, specifically including the Unified Process;
CO6: properly utilized a modern CASE tool environment, specifically including UML modeling;
CO7: developed and executed a plan for product testing and evaluation;
CO8: participated in the development and presentation of group projects;
CO9: demonstrated the ability to critically analyze materials ranging from project proposals to scholarly research and to express this analysis clearly in both spoken and written form.
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 48.5. This course has approximately 40 lecture hours of student contact plus approximately 40 laboratory hours of student contact. Some of the 48.5 hours are covered during scheduled lab, which students are required to attend. Several topics, particularly IM1: information systems and models; IM2 database systems; IM3: data modeling; SE3: software tools and environments, and SE8: software project management, 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:

- **Scope of Software Engineering**
  - historical and economic aspects of Software Engineering
  - "aspect" vs. "phase" - uncoupling of tasks from when or where the tasks occur
  - aspects: maintenance, requirements, analysis and design, implementation
  - social context of software development
    - ACM & IEEE Code of Ethics
    - ethical issues

- **Software Life Cycle Models**
  - code and fix
  - waterfall
  - rapid prototyping
  - extreme programming, agile processes
  - synchronize and stabilize
  - spiral
  - iteration and incrementation

- **Software Development Process**
  - Unified Process
  - Object-Oriented paradigm vs. Structured Programming paradigm
  - workflows
    - requirements
    - analysis
    - design
    - implementation
    - test
  - post-delivery maintenance
  - retirement
  - phases of the Unified Process: inception, elaboration, construction, transition
  - one vs. two dimensional file cycle models
  - Capability Maturity Models (CMM)
  - ISO standards

- **Software Development Teams**
  - team organization paradigms
    - democratic teams
    - classic chief programmer-led teams
    - Synchronize and Stabilize teams
    - Extreme Programming teams
  - choosing an appropriate organization
  - social and ethical issues

- **Tools of the trade**
  - conceptual tools
    - stepwise refinement
    - cost-benefit analysis
    - software metrics
  - CASE tools
• taxonomy
  • scope
    o configuration and version control systems
    o build tools
    o the role of CASE technology in software development
    o social and ethical issues
• Testing  
  SE2(0.5),SE3(0.5),SE5(0.5),SE6(1),SE11(0.5), SP5(0.5),
  • quality issues
    o engineering definition of quality
    o user definition of quality
    o software quality assurance
  • non-execution-based testing
    o walkthroughs
    o inspections
  • execution-based testing
    o black-box, white-box (clear-box)
    o regression testing
  • focus of testing: utility, reliability, robustness, performance, correctness
  • correctness proofs
  • role of quality assurance in testing
  • social and ethical issues
• Modules and Object Design  
  PL5(0.5), SE1(0.5)
  • definition of a module
  • cohesion
  • coupling
  • encapsulation of data
  • Abstract Data Types (ADTs)
  • information hiding
  • object-oriented vs. non-object-oriented design
  • inheritance
  • polymorphism
• Resuability  
  SE1(0.5),SE7(0.5), SP3(0.5),SP4(0.5),SP5(0.5),SP6(0.5),SP9(0.5)
  • reuse concepts and degrees
  • impediments to reuse - what makes code hard to re-use
  • objects in the context of reuse
  • reuse during design and implementation
    o application frameworks
    o design patterns
    o software architecture
    o component-based software engineering
  • social and ethical issues
• Portability  
  SE2(0.5),SE9(0.5),SE12(0.5)
  • hardware and operating system incompatibilities
  • techniques for addressing portability needs
• Planning and Estimating  
  SE8(1.5)
  • estimating time and cost
    o metrics for size
    o techniques for cost estimation
    o COCOMO, COCOMO II
    o tracking estimates
  • project management plan components
    o IEEE Software Project Management Plan
  • planning testing
  • establishing documentation standards
  • CASE tools for Planning and Estimating
  • testing and evaluating the software project management plan

Page 3 of 5
The emphasis of the course is on state-of-practice methodologies and patterns, the application of which will result in cost-effective and efficient development of software systems that meet user needs and expectations and are also flexible and maintainable. 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. Weekly writing assignments based on assigned articles and internet research will serve to broaden students’ exposure to recent developments in the field and to social and ethical aspects of software engineering.

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: 20%; presentations: 10%; midterm and final exam: 40%; homework: 10%; papers: 20%.

<table>
<thead>
<tr>
<th></th>
<th>Exam Questions</th>
<th>Homework Problems</th>
<th>Programming Projects</th>
<th>Lab Exercises</th>
<th>Papers</th>
<th>Group Projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO01</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>
</tr>
<tr>
<td>CO03</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>
</tr>
<tr>
<td>CO05</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>
</tr>
<tr>
<td>CO07</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CO08</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CO09</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Web Resources:
- Association for Computing Machinery (ACM). http://www.acm.org/
- The Institute of Electrical and Electronics Engineers (IEEE). http://www.ieee.org/portal/site

Bibliography:
- Dikel, David M.; Kane, David; Wilson, James R. Software Architecture: Organizational Principles and Patterns. Prentice-Hall, 2001
Gamma, Helm, Johnson & Vlissides. Design Patterns. Addison-Wesley, 1995
Larman, Craig. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. Pearson Education, 2005
Quinn, Michael. Ethics for the Information Age. Addison-Wesley, 2005
Shalloway, Alan; Trott, James. Design Patterns Explained: A New Perspective on Object-Oriented Design. Addison-Wesley, 2002