COURSE DESCRIPTION

<table>
<thead>
<tr>
<th>Dept., Number</th>
<th>Course Title</th>
<th>Software Engineering: Design and Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS 4311</td>
<td>Required</td>
<td></td>
</tr>
<tr>
<td>Semester hours</td>
<td>45 hours</td>
<td>Course Coordinator: Yoonsik Cheon and Ann Gates</td>
</tr>
</tbody>
</table>

Current Catalog Description
Methodologies, approaches, and techniques associated with software design, implementation, and testing of a software system; other topics include cooperative teamwork, project management, and documentation; second semester of a two semester capstone project in which students design and implement a real-world application specified in CS 4310.

Textbook:

Other References

Course Outcomes:

**Level 1: Knowledge and Comprehension**
Upon successful completion of this course, students will be able to:
1a. Articulate design principles, including cohesion and coupling, encapsulation, and information hiding.
1b. State the main features of PSP.
1c. Relate the main factors in cost estimation.
1d. Discuss the issues related to maintenance.
1e. Discuss different software architectural styles such as blackboard, event systems, layered system, pipe and filters.
1f. Describe the main components of the Software Engineering Code of Ethics.

**Level 2: Application and Analysis**
Upon successful completion of this course, students will be able to:
2a. Differentiate between audits, inspections, and walk-throughs.
2b. Apply different diagramming techniques for an architectural design, including class diagrams, CRC cards, module diagrams, structure charts, and subsystem diagrams.
2c. Apply different diagramming techniques for producing a detailed design of a system, including collaboration diagrams, decision tables and diagrams.
2d. Relate general strategies to identify and implement appropriate software architecture patterns according to the system being developed.
2e. Relate general strategies for creating a design of a system.
2f. Distinguish between the different levels of cohesion and coupling.
2g. Apply project management techniques.
2h. Use software development and maintenance tools (creation and editing of software documents), GUI generators, comprehension tools, supporting activities tools (configuration management tools), and verification and validation tools.

2i. Demonstrate skills in working as a cooperative team in order to achieve group goals.

2j. Examine potentially negative technical, ethical, or legal implications of software engineering projects on customers or others involved.

2k. Exhibit responsible attitudes and work habits as individuals and groups, in accordance with professional software engineering codes of ethics.

2l. Relate the importance of professional societies.

2m. Describe the difference between unit, integration, system, and acceptance testing.

2n. Engage in self-directed study to learn new techniques and tools for software design, implementation, and/or testing.

**Level 3: Synthesis and Evaluation**

Upon successful completion of this course, students will be able to:

3a. Conduct a technical review.

3b. Implement a configuration management plan.

3c. Create an architectural design and detailed design for a large system.

3d. Develop a design that anticipates change.

3e. Construct software from a software design.

3f. Develop white-box and black-box test cases that provide a variety of test coverage.

3g. Conduct unit and integration testing.

3h. Develop a test plan for a large system.

3i. Demonstrate an ability to assemble and orally present technical work to different constituencies.

3j. Compose technical documents that are grammatically correct and technically sound.

3k. Apply effective techniques for collaboration and problem-solving within groups.

**Student Outcomes**

Student Outcomes: 1, 2, 3, 4, 5, 6, 7, 8, 9, 11

**Prerequisites by Topic**

CS 4310 with a grade of C or better

**Major Topics Covered**

- Configuration management
- Software design analysis techniques
- High-level software design
- Software design specification
- Software implementation
- Software validation and verification
- Software process improvement