95-706: Object-Oriented Software Analysis and Design
Instructor: C. Aaron Cois, PhD

Course Information

Description:
Large-scale software development has been described as one of the most difficult of human undertakings. This course examines the reasons for the inherent complexity of software construction, and presents structured methods to deal effectively with it. The course will focus on the object-oriented approach for analysis and design. Students will gain an appreciation of the difference between writing programs and doing analysis and design. Problem formulation and decomposition (analysis) and solution building (design) will be covered. Students will work in small groups, each group having the responsibility for analysis, design and implementation of a software system. Case tools will be used in several stages of the development process.

Learning Objectives:

• Develop a working understanding of formal object-oriented analysis and design processes.
• Identify and describe the primary risks inherent to large-scale software development.
• Apply techniques and processes designed to mitigate these risks. Construct OOAD artifacts designed to enable these processes.
• Develop the skills to determine which processes and OOAD techniques should be applied to a given project.
• Prepare students for industry by developing an understanding of the application of OOAD practices from both software project management and software development perspectives.

Prerequisites:
Knowledge of an Object-Oriented language such as Java, Python, or C++ is required for this course.

Administrative Topics

Grading:
Grading is project-based. There will be 5 projects assigned, 3 of which will be team projects. Participating in projects and project teams is mandatory, and projects all build on the same case study as the term progresses. Missing time, even early in the term, will be severely detrimental to overall project performance, and thus course grade. If a student is unable to avoid joining the class late and has missed assignment(s), the weight of these assignments on their final grade will be evenly distributed over all other assignments.
**Assignment Submission:**

Assignments are due at the beginning of class on due date, unless otherwise specified. Unless otherwise specified, a paper copy of each assignment must be submitted at the deadline. **NO LATE SUBMISSIONS** will be accepted. No exceptions will be made.

**Policy on grading disputes:**

When questions arise in grading of a project, the following steps must be taken to address the issue:

1. The student(s) will speak personally with the TA who graded the assignment, and allow the TA to explain the reason points were taken off.
2. If the student(s) still disagree with the deduction, they will write a clear, detailed email to the professor (CC’ing the TA) explaining why they feel points should not have been deducted.
3. The TA will then respond to the professor only with their perspective on the grading, and the professor will make a final decision on whether points will be returned.

This procedure is mandatory to address grading issues.

**Lectures:**

Lecture material is supplemental to material covered in assigned readings, both textbook and case study. You are responsible for all information covered in lectures, as well as information presented in posted slides and course readings. Being absent from lecture, for any reason, is no excuse for not knowing presented information.

**Grading Scale:**

<table>
<thead>
<tr>
<th>Percentage</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>97.5-100</td>
<td>A+</td>
</tr>
<tr>
<td>92.5-97.4</td>
<td>A</td>
</tr>
<tr>
<td>90.0-92.4</td>
<td>A-</td>
</tr>
<tr>
<td>87.5-89.9</td>
<td>B+</td>
</tr>
<tr>
<td>82.5-87.4</td>
<td>B</td>
</tr>
<tr>
<td>80.0-82.4</td>
<td>B-</td>
</tr>
<tr>
<td>77.5-79.9</td>
<td>C+</td>
</tr>
<tr>
<td>72.5-77.4</td>
<td>C</td>
</tr>
<tr>
<td>70.0-72.4</td>
<td>C-</td>
</tr>
</tbody>
</table>

...and so on...
Schedule

**January 14**

**Topics:**
- Introduction: What is OOAD?
- Software Development Life Cycle (SDLC) & Rational Unified Process (RUP)
- Requirements Elicitation & Analysis
- Quality Attributes
- Assignment of Project Teams

**Reading for this week:**
- Case Study: “Why Software Fails”, IEEE 2005 (available on Blackboard)
- Karl E. Wiegers, Chapter 5, “Establishing the Project Vision and Scope”
- Karl E. Wiegers, Chapter 6, “Finding the Voice of the Customer”
- Karl E. Wiegers, Chapter 8, “Understanding User Requirements”
- Karl E. Wiegers, Chapter 10, “Documenting the Requirements”
- Karl E. Wiegers, Chapter 12, “Beyond Functionality: Software Quality Attributes”

**Assignments:**

**January 21**

**Topics:**
- More on Requirements Analysis
- Case Study: POST
- Use Cases

**Reading for this week:**
- Karl E. Wiegers, Chapter 5, “Establishing the Project Vision and Scope”
- Craig Larman, Chapter 3, “Case Studies”
- Craig Larman, Chapter 6, “Use Cases”
- Craig Larman, Chapter 30, "Relating Use Cases”

**Assignments:**
• Assignment 1: Question and Assumption lists in preparation for Requirements Elicitation, Due. Sunday, January 24 by 11:59pm. via Blackboard submission.
• Team Requirements Elicitation sessions performed as scheduled with TA “Customer” by next lecture (1/28)

January 28

Topics:

• From Use Cases to Functional Requirements
• Domain Model
• Associations
• Attributes
• Generalization

Reading for this week:

• Craig Larman, Chapter 9, “Domain Models”
• Craig Larman, Chapter 31, “Domain Model Refinement”

Assignments:

• Begin Assignment 2: Software Requirements Specification, Use Cases, and Project Lexicon document

February 4

Topics:

• System Behavior
• System Sequence Diagrams
• Contracts
• State Diagrams
• Interaction Diagrams

Reading for this week:

• Craig Larman, Chapter 10, “System Sequence Diagrams”
• Craig Larman, Chapter 11, “Operation Contracts”
• Craig Larman, Chapter 32, “More SSDs and Contracts”
• Craig Larman, Chapter 29, “UML State Machine Diagrams and Modeling”
• Craig Larman, Chapter 28 “UML Activity Diagrams and Modeling”
• Craig Larman, Chapter 17, “GRASP: Designing Objects with Responsibility”
Assignments:

- Assignment 2 Due at the beginning of class
- Begin Assignment 3: Domain Model

February 11

Topics:

- GRASP
- Determining Visibility
- Design Class Diagrams
- Law of Demeter

Reading for this week:

- Craig Larman, Chapter 17, “GRASP: Designing Objects with Responsibility”
- Craig Larman, Chapter 18, “Object Design Examples with GRASP”
- Craig Larman, Chapter 25, “GRASP: More Patterns for Assigning Responsibility”
- Craig Larman, Chapter 19, “Designing for Visibility”
- Craig Larman, Chapter 16, “UML Class Diagrams”

Assignments:

- Assignment 3 Due at the beginning of class
- Begin Assignment 4: Class Diagram

February 18

Topics:

- Class Design Principles
- Class Design Issues
- GoF Design Patterns

Reading for this week:

- Craig Larman, Chapter 26, “Applying GoF Design Patterns”
- Robert C. Martin, “Design Principles and Design Patterns”
• GoF, “Introduction to Design Patterns,” (Chapter 1 of E. Gamma, R. Helm, R. Johnson, J, Vlissides, “Design Patterns”, Addison-Wesley, 1994, pp. 1--31)

Assignments:

• Keep working on Assignment 4

February 25

Topics:

• More GoF Design Patterns
• System Design/Architecture
• OO Metrics

Reading for this week:

• GoF, “Introduction to Design Patterns,” (Chapter 1 of E. Gamma, R. Helm, R. Johnson, J, Vlissides, “Design Patterns”, Addison-Wesley, 1994, pp. 1--31)
• Chidamber & Kemerer: “A Metrics Suite for O-O Design,” (IEEE TSE `94)

Assignments:

• Assignment 4 Due at the beginning of class
• Assignment 5 Due Sunday February 28, 11:59pm via Blackboard Submission.

March 3

Graded Assignment(s) 1-4 available for pickup

March 10

Final Grades Due