# End of Course Memo
# CS 494 – Object Oriented Design
# Aaron Bloomfield (Spring 2006)

## *Course Objectives:*

1. Create a requirements model using UML class notations and use-cases based on statements of user requirements, and to analyze requirements models given to them for correctness and quality.
2. Create the OO design of a system from the requirements model in terms of a high-level architecture description, and low-level models of structural organization and dynamic behavior using UML class, object, and sequence diagrams.
3. Comprehend enough Java to see how to create software the implements the OO designs modeled using UML.
4. Comprehend the nature of design patterns by understanding a small number of examples from different pattern categories, and to be able to apply these patterns in creating an OO design.
5. Given OO design heuristics, patterns or published guidance, evaluate a design for applicability, reasonableness, and relation to other design criteria.

## *Assessment of Learning by Course-Objective:*

**Objective 1: [Create a requirements model using UML class notations and use-cases based on statements of user requirements, and to analyze requirements models given to them for correctness and quality.]**
The lectures discussed these topics, and a number of the homeworks required UML class notations and use cases. This material was also tested on the exams.

**Objective 2: [Create the OO design of a system from the requirements model in terms of a high-level architecture description, and low-level models of structural organization and dynamic behavior using UML class, object, and sequence diagrams.]**
This material was discussed at length in lecture. The five homeworks, which were all part of a single course-wide project, required a lot of UML diagrams, as well as the overall OO design. Thus, the students learned the theory in lecture and applied that theory during the homeworks.

**Objective 3: [Comprehend enough Java to see how to create software the implements the OO designs modeled using UML.]**
This requirement was modified a bit during the course – instead of 'Java', the course ensured that the students knew multiple high-level OOP languages (it is possible a student could complete this course and still not know Java – but as the idea of this course is to ensure they understand high-level OO design, the spirit of this requirement was met). The students had to do their homeworks in a OOP language that they did NOT already know

**Objective 4: [Comprehend the nature of design patterns by understanding a small number of examples from different pattern categories, and to be able to apply these patterns in creating an OO design.]**
The lectures discussed these topics. Thus, the students learned this objective through the lectures, and were assessed on this on the midterm and final exam. The homeworks required that they apply 10 different design patterns to their course project.

**Objective 5: [Given OO design heuristics, patterns or published guidance, evaluate a design for applicability, reasonableness, and relation to other design criteria.]**
This was discussed in lecture – a number of case studies were presented, and the various design decisions were analyzed in detail.

## *Assessment of Changes Made in the Course:*

As this was my first semester teaching this course, I did not make many significant changes to the course. The few changes I did make are listed below.

- There were 5 homeworks in this course, and together they constituted the course project. Each homework was a separate iteration of the same software.
- The students were allowed to choose what they wanted their course project to be. A number chose games, but there were other ideas as well – social networking software, museum management software, web/DB interface system, etc. This allowed for the students to be more invested in their projects.
- The students were allowed to do their homeworks in groups (generally, groups of 2, but I allowed for two groups of 3).
- The course project could be in any language, as long as it met two requirements: (1) it was an OO language (obviously), and (2) it was a language that they did NOT already know. This last requirement was met with some resistance from the students, but I felt it was a good requirement – it forced them to learn a new language.
- A new textbook was used ("Applying UML and Patterns", by Craig Larman). In future iterations of this course, a different textbook will be used.

## *Other Issues:*

1. Do you have concerns regarding the background of students coming into the course?

   None. I assumed that they knew a single OO language, and general OOP principles, and this was the case.

2. Are there other issues affecting student learning beyond what has been discussed elsewhere in this report? Include any other concerns you have about what students have or have not learned when they have completed the course.

   None.

3. If you know of changes being made or considered in the curriculum that might affect the course, briefly describe what these are and how the course might be affected.

   None. This course is pretty self-contained, and just requires a knowledge of a single OO language (and how to program in OO, of course). So this cousre can adapt to any curriculum changes quite easily.

4. List any other comments you think the Committee that monitors our degree programs should know about this course this semester.

   None.

## ***Mapping of Course Objectives to BSCS Outcomes:***

| **CS Degree Outcomes: Students who graduate with a BSCS will…** | **Course Obj. 1** | **Course Obj. 2** | **Course Obj. 3** | **Course Obj. 4** | **Course Obj. 5** |
|---|---|---|---|---|---|
| (1: Math & DLD) Have demonstrated comprehension in relevant areas of mathematics (including calculus, discrete math, and probability), and in the area of logic design. | | | | | |
| (2: Fundamentals) Have demonstrated comprehension in fundamental topics of computing, including the intellectual core of computing, software design and development, algorithms, computer organization and architecture, and software systems. | | | X | | |
| (3: Analysis & Evaluation) Have applied knowledge of areas of computing to analyze and evaluate algorithms, designs, implementations, systems, or other computing artifacts or work-products. Application of this knowledge includes the ability to design, conduct and evaluate the results of experiments and testing activity. | X | X | X | X | X |
| (4: Build Solutions) Have applied knowledge of areas of computing to create solutions to challenging problems, including specifying, designing, implementing and validating solutions for new problems. | X | X | X | X | X |
| (5: Research Awareness) Be aware of current research activity in computing through activities including reading papers, hearing research presentations, and successfully planning and completing an individual research project in computing or its application. | | | | | |
| (6: Broadening) Have demonstrated comprehension of subjects in the humanities, social sciences, and the natural sciences in order to broaden a student's education beyond engineering and computing. | | | | | |
| (7: Social and Professional) Comprehend important social, ethical, and professional considerations related to computing practice and research, and be able to apply this knowledge when analyzing new situations. | | | | | |
| (8: Post-graduation) Be prepared to enter graduate programs in computing or related fields, and be prepared to begin a professional career in computing. | | | | | |
| (9: Life-long Learning) Have demonstrated a self-directed ability to acquire new knowledge in computing, including the ability to learn about new ideas and advances, techniques, tools, and languages, and to use them effectively; and to be motivated to engage in life-long learning. | | | | | |
| (10: Teamwork) Have demonstrated the ability to work effectively in a development team. | X | X | | | |
| (11: Communication) Have demonstrated the ability to communicate effectively (orally and in writing) about technical issues. | X | X | | | |
| (12: Professional development practices) Comprehend important issues related to the development of computer-based systems in a professional context using a well-defined process to guide development. | X | X | | | |