Course Outcomes	CSC 3380

CSC 3380: Object Oriented Design

Credit Hours: 3 hours

Prerequisites:
> CSC 1254 or CSC 1351.

Prerequisites by Topics:
> Any 2-course programming sequence in an object-oriented language

Catalog Course Description:
> Advanced object oriented software development. Use of the unified modeling language as a design tool will be emphasized.

Course Outcomes
1. Master the fundamental principles of OO programming,
2. Master key principles in OO analysis, design, and development ,
3. Be familiar with the application of the Unified Modeling Language (UML) towards analysis and design,
4. Master common patterns in OO design and implement them,
5. Be familiar with alternative development processes,
6. Be familiar with group/team projects and presentations.
7. Be exposed to technical writing and oral presentations.

Texts and Other Course Materials

Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. Design Patterns: Elements of Reusable Object- Oriented Software . ISBN 0-201-63361-2 HB 1995 Addison.

References:
Kent Beck. Test-Driven Development: By Example. Addision Wesley, Reading, MA. 2003. ISBN 0-321-14653-0.

Martin Fowler. Refactoring: Improving the Design of Existing Code. Addison Wesley, Reading MA. 1999. ISBN 0-201-48567-2.

Martin Fowler and Kendall Scott. UML Distilled: A Brieg Guide to the Standard Object Modeling Language, 2nd Ed. Addison Wesley, Reading MA. 1999. ISBN 0-320-16578-3X.

Joshua Kerievshy. Refactoring To Patterns. Addision Wesley, Reading, MA. 2005. ISBN: 0-3212-1335-1.

Robert C. Martin. <u>Agile Software Development: Principles, Patterns and Practices</u>. Prentice Hall, Upper Saddle River, NJ. 2003. ISBN: 0-135-97444-5.

Major Topics

- Building quality software systems
- Introduction to alternative OO development processes: test-driven development, pair programming, agile development
- The Agile Manifesto
- User stories and system specification
- Unit testing and test-driven development
- Pair programming
- Refactoring and design improvement
- General introduction into Universal Modeling Language (UML): structure diagrams (class, object, and component diagrams), behavior diagrams (activity diagrams, state charts, use cases, sequence diagrams, communication diagrams)
- Design Patterns: their intent, applicability (including benefits and drawbacks), structure and implementation
- Advanced object-oriented programming needed in implementation of certain patterns: in Java (dynamic class loading, static blocks, interfaces, inner class tricks) and in C++ (abstract base classes, virtual functions, private and public multiple inheritance mixtures)
- Inter-personal dynamics that occur within a small project team

Assignments/Projects/Laboratory Projects/Homework

- Several small applications involving practice with design patterns and XP techniques
- Term Project: medium-sized software system with deliverables and presentations

Oral Communication (Presentations)

The term project will be a medium-sized group project involving the design and development of a non-trivial software system; the class will be divided into one or more working groups, with students being strongly encouraged to form their own based upon compatability.

There will be some oral presentations, including a final project demonstration. All group members must participate in the final presentation; each will be judged and graded on preparation and presentation skills as well as content.

Written Communication

As an essential part of the term project each of the groups are required to submit a series of documents, minimally including a functional specification and a general design document. All group members must contribute to composition of these papers; they will be judged and graded on effective writing style and grammatical correctness as well as content.

Curriculum Category Content (estimated in semester hours)

Area	Core	Advanced	Area	Core	Advanced
Algorithms	10	11	Data Structures		
Software Design	15	8	Prog. Languages		
Computer Arch.					

Relationship to Criterion 3 Outcomes

A	B	C	D	E	F	G	H	I	J	K
*		*	*	*		*		*		

Math and Fundamentals:

Data Structures:

Algorithms and Software:
UML – 5 hours, Design patterns – 26 hours, Agile programming foundations – 6 hours, Refactoring 2-hours, test-driven development – 5 hours,
Computer Organization and Architecture:

Concepts of Programming Languages:

Social and Ethical Issues:

Oral Communication (presentations)
Presentations – 3 hours

Written Communication:

Course Coordinator: S. Kundu
Last Modified: September 11, 2009