UNIVERSITY OF NEBRASKA AT OMAHA
COURSE SYLLABUS/DESCRIPTION

Department and Course Number	CSCI 4970
Course Title	Computer Science Capstone
Course Coordinator	H. Siy
Total Credits	3
Date of Last Revision	March 3, 2014

1.0 Course Description

1.1 Overview of content and purpose of the course (Catalog description).
The Capstone Project completes a Computer Science student's undergraduate experience. Students will work on a team-based real-world project, practicing software engineering skills and applying fundamental computer science principles acquired throughout their undergraduate study.

1.2 Prerequisites of the course.
CSCI 4830. Senior standing in Computer Science.

1.3 Overview of content and purpose of the course.
This course serves to round out a Computer Science student's undergraduate experience by enabling him or her to apply fundamental computer science principles to the solution of real-world problems. Furthermore, the student gets to apply software development skills on a large project, working together in groups. Such an experience is intended to help prepare the student for the career realities of a software professional.

As this is a project-oriented course, the content will vary from project to project. Some topics that may be covered include:
Team collaboration strategies
Version control
Requirements modeling
Design modeling
Software analysis and testing

1.4 Unusual circumstances of the course.
None.

1.5 For whom course is intended.
All students majoring in Computer Science.

2.0 Course Justification Information

2.1 Anticipated audience / demand:

This is a required course for all undergraduate students majoring in Computer Science.

2.2 Indicate how often this course will be offered and the anticipated enrollment:

Every semester

2.3 If it is a significant change to an existing course, please explain why it is needed:

n/a

3.0 Objectives

3.1 List of performance objectives stated in terms of the student educational outcomes.

- 3.1.1 Apply fundamental computer science principles in solving real-world problems.
- 3.1.2 Develop communication skills through interaction with client.
- 3.1.3 Gain proficiency in modeling, implementing and testing large software applications.
- 3.1.4 Learn to work in teams.

4.0 Content and Organization

List of major topics to be covered in chronological sequence (specify number of contact hours on each).

As this is a project-oriented course, the content will vary from project to project. Some topics that may be covered include:

- 4.1 Team collaboration [3]
  - 4.1.1 Group dynamics
  - 4.1.2 Geographically distributed software development
- 4.2 Modeling requirements and solutions [3]
  - 4.2.1 Requirements elicitation and analysis
  - 4.2.2 Architecture and design
- 4.3 Source code analysis [3]
  - 4.3.1 Reasoning about programs
  - 4.3.2 Version control
- 4.4 Testing [3]
  - 4.4.1 Test case development
  - 4.4.2 Regression testing

Additional project-specific lectures may be provided by the instructor and/or client.

5.0 Teaching Methodology Information

5.1 Methods:

This is mainly a project-oriented course. The instructor will function as a consultant to the project and will monitor the progress of the project.

Lectures reviewing the software development process will be provided.

Additional lectures specific to a project topic may also be provided.

5.2 Student role:

Students will form software development teams.

Students will work with a client to develop a project proposal.

Once the proposed project has been approved by the instructor and client, students will consult with the instructor and develop a project plan for delivering a final product acceptable to the client.

Students will be accountable for meeting the planned milestones and reporting progress to the instructor and/or the client.

Upon completion of the project, students will publicly present and demonstrate their work.

6.0 Evaluation Information

6.1 Describe the typical types of student projects that will be the basis for evaluating student performance:

Projects are expected to have a challenging software component requiring the nontrivial application of computer science principles. Example application types include scheduling, simulation, data mining, cyber-physical systems, and scientific computing. Projects will usually involve sophisticated algorithms manipulating complex structures, and may involve overlapping areas such as language processing, networks, embedded systems, real-time systems, parallel computation, databases, artificial intelligence, computer graphics, etc.

6.2 Describe the typical basis for determining the final grade (e.g. weighting of various student projects):

The course grade is based on the successful completion of the project, the quality of the project artifacts, and strength of individual contributions.

Individuals will be evaluated based on each person's performance on the project. Individual project contribution will be assessed based on weekly log activities, quality of contributions, and peer evaluations.

6.3 Grading type:

Points	Grade
97-100%	A+
93-96%	A
90-92%	A-
87-89%	B+
83-86%	B
80-82%	B-
77-79%	C+
73-76%	C
70-72%	C-
67-69%	D+
63-66%	D

60-62%	D-
0-59%	F

7.0 Resource Material

7.1 Textbooks and/or other required readings used in course

7.1.1 F. Brooks, *The Mythical Man-Month: Essays on Software Engineering, 2nd Edition*, Addison-Wesley, 1995.

7.1.2 B. Liskov and J. Guttag, *Program Development in Java: Abstraction, Specification, and Object-Oriented Design*, Addison-Wesley, 2000.

7.1.3 I. Sommerville, *Software Engineering, 9th edition*, Addison-Wesley, 2010.

7.2 Other suggested reading materials, if any

7.2.1 F. Brooks, *The Design of Design: Essays from a Computer Scientist*, Addison-Wesley, 2010.

7.2.2 B. Bruegge and A. Dutoit. *Object-Oriented Software Engineering, 2nd edition*, Prentice Hall, 2004.

7.2.3 B. Chess, *Secure Programming with Static Analysis*, Addison-Wesley, 2007.

7.2.4 A. Cooper, *About Face 3.0*, Wiley, 2007.

7.2.5 R. Graham, D. Knuth, O. Patashnik. *Concrete Mathematics: A Foundation for Computer Science, 2nd edition*, Addison-Wesley, 1994.

7.2.6 D. Norman, *The Design of Everyday Things*, Basic Books, 2002.

7.2.7 M. Pezze and M. Young. *Software Testing and Analysis: Process, Principles and Techniques*, Wiley, 2008.

7.3 Other sources of information.

Technology- and domain-specific information to be provided by clients.

7.4 Current bibliography of resource for student's information

7.4.1 F. Brooks. No Silver Bullet – Essence and Accidents of Software Engineering. *IEEE Computer*, 20(4): 10-19, April 1987.

7.4.2 Melvin Conway. How do Committees Invent? *Datamation* 14(4): 28–31, 1968.

7.4.3 M. E. Fagan. Design and code inspections to reduce errors in program development. *IBM Systems Journal*, 15(3), 1976.

7.4.4 A. Meneely and L. Williams. On Preparing Students for Distributed Software Development with a Synchronous, Collaborative Development Platform. *ACM Technical Symposium on Computer Science Education (SIGCSE)*, 2009.

7.4.5 Eric Raymond. The Cathedral and the Bazaar. *Linux Kongress*, May 1997, http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar. (Last accessed March 18, 2010)

Additional reference material may be required based on actual project.

8.0 Computer Science Accreditation Board (CSAB) Category Content (class time in hours)

CSAB Category	*Core*	*Advanced*
Data structures		10
Computer organization and architecture		
Algorithms and software design		10
Concepts of programming languages		

9.0 Oral and Written Communications

Students will develop written requirements and analysis and design models to be presented to the client for validation. They will also be required to make a presentation and demonstration of their final project.

10.0 Social and Ethical Issues

Covered by prerequisite.

11.0 Theoretical content

	Contact hours
Group dynamics	1.0
Version control	1.0
Program analysis	4.0

12.0 Problem analysis

Students learn to elicit requirements from actual clients and formulate them into a software problem.

13.0 Solution design

Students learn to implement the requirements systematically, through refinement of models. Students learn about relevant technologies and apply them to the problem. Students learn to create regression tests to make sure that existing services that should not change are left unchanged. Students learn to use version control as a way of controlling changes to the system.

CHANGE HISTORY

Date	***Change***	***By whom***	***Comments***
02/19/2009	Initial version	Siy	
02/19/2009	Modified sections 1, 3, 4 and 6 after discussion during UPC meeting. Also fixed the information in sections 8-12.	Siy	
02/27/2009	Modified as a special topics proposal for Fall 2009.	Siy	
03/20/2010	Modified description to make it suitable as a capstone course.	Siy	
03/30/2010	Modified the wording to clarify that all team projects are acceptable as long as they solve a Computer Science-related problem.	Siy	
04/01/2010	Clarified that the proposed project must be approved by instructor and client.	Siy	
03/03/2014	Alphabetized references.	Siy	

UNIVERSITY OF NEBRASKA AT OMAHA
Mapping of CS Program Outcomes vs. course objectives

Department and Course Number	CSCI 4960
Course Title	Capstone Project
Course Coordinator	Harvey Siy
Total Credits	3
Date of Last Revision	April 1, 2010

Instructions: Paste or type the course objectives in the left-hand column. Indicate the relationship between course objective and program outcome by placing one of the two following marks in the appropriate cell:

S – Strong relationship
X – Contributing relationship

Course objective	**CS Program Outcomes**										
	(a) knowledge of discipline	**(b) analyze problem, define**	**(c) design and implement solution**	**(d) function on a team**	**(e) ethical issues**	**(f) communicate effectively**	**(g) analyze impact of computing**	**(h) continued professional development**	**(i) Current techniques and tools**	**(j) apply foundations**	**(k) apply design and development**
1. Apply fundamental computer science principles in solving real-world problems.	**S**	**S**								**S**	
2. Develop communication skills through interaction with client.					**X**	**S**	**S**	**S**			
3. Gain proficiency in modeling, implementing and testing large software applications.		**S**	**S**			**X**	**X**		**S**		**S**
4. Learn to work in teams.				**S**	**X**	**S**			**X**		

CS Program Outcomes (2008)

(a) An ability to apply knowledge of computing and mathematics appropriate to the discipline;
(b) An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution;
(c) An ability to design, implement and evaluate a computer-based system, process, component, or program to meet desired needs;
(d) An ability to function effectively on teams to accomplish a common goal;
(e) An understanding of professional, ethical, legal, security, and social issues and responsibilities
(f) An ability to communicate effectively with a range of audiences
(g) An ability to analyze the local and global impact of computing on individuals, organizations and society
(h) Recognition of the need for, and an ability to engage in, continuing professional development
(i) An ability to use current techniques, skills, and tools necessary for computing practices
(j) An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices
(k) An ability to apply design and development principles in the construction of software systems of varying complexity.