UNIVERSITY OF NEBRASKA AT OMAHA
COURSE SYLLABUS

Department and Course Number	CSCI 4830
Course Title	Introduction to Software Engineering
Course Coordinator	Harvey Siy
Total Credits	3
Repeat for Credit?	No
Date of Last Revision	March 3, 2014

1.0 Course Description Information

1.1 Catalog description:

Basic concepts and major issues of software engineering, current tools and techniques providing a basis for analyzing, designing, developing, maintaining and evaluating software systems. Technical, administrative and operating issues. Privacy, security and legal issues.

1.2 Prerequisites of the course:

1.2.1 CSCI 3320
1.2.2 Familiar with advanced data structures
1.2.3 Familiar with main concepts of analysis of algorithms
1.2.4 Concepts of structured problem solving and programming
1.2.5 Concepts of object-oriented problem solving programming
1.2.6 Proficiency in one of the modern programming languages

1.3 Overview of content and purpose of the course:

This course covers topics on development of software systems. It provides students with knowledge of performing system and software requirement analysis and specification, architecture and detailed design, testing, and integration techniques. It also presents the basics of project management and object oriented methodologies.

1.3.1 Software project management
1.3.2 Software life cycle and process
1.3.3 Requirement analysis
1.3.4 System and information engineering
1.3.5 Analysis and design methods

1.4 Unusual circumstances of the course.

none

2.0 Course Justification Information

2.1 Anticipated audience / demand:

The course is intended for upper division undergraduate CS or MIS majors who wish to pursue the topic of Engineering and development of software systems.

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 List of performance objectives stated in learning outcomes in a student's perspective:

3.1 Perform analysis and design of small and medium-sized software project using structured methods.
3.2 Be able to participate in design of small and medium-sized software project using object-oriented software development methodologies.
3.3 Prepare software project management documents.
3.4 Be able to participate in a project team.
3.5 Develop parts/whole prototype as well as implementation of small or medium-sized software projects.
3.6 Introduce socio-technical and ethical issues in the development of real-world software systems.

4.0 Content and Organization Information

4.1 List the major topics central to this course:

4.1.1 Introduction
- 4.1.1.1 FAQs about software engineering
- 4.1.1.2 Professional and ethical responsibility

4.1.2 Computer-based System Engineering*
- 4.1.2.1 Emergent system properties
- 4.1.2.2 Systems and their environment
- 4.1.2.3 System modeling
- 4.1.2.4 The system engineering process
- 4.1.2.5 System procurement

4.1.3 Software Processes
- 4.1.3.1 Software process models
- 4.1.3.2 Process iteration
- 4.1.3.3 Software specification
- 4.1.3.4 Software design and implementation
- 4.1.3.5 Software validation
- 4.1.3.6 Software evolution
- 4.1.3.7 Automated process support

4.1.4 Project Management
- 4.1.4.1 Management activities
- 4.1.4.2 Project planning
- 4.1.4.3 Project scheduling
- 4.1.4.4 Risk management

4.1.5 Managing People*
- 4.1.5.1 Limits to thinking
- 4.1.5.2 Group working
- 4.1.5.3 Choosing and keeping people
- 4.1.5.4 The people capability maturity model

4.1.6 Software Cost Estimation*
- 4.1.6.1 Productivity
- 4.1.6.2 Estimation techniques
- 4.1.6.3 Algorithmic cost modeling
- 4.1.6.4 Project duration and staffing

4.1.7 Quality Management*
- 4.1.7.1 Quality assurance and standards
- 4.1.7.2 Quality planning
- 4.1.7.3 Quality control
- 4.1.7.4 Software measurement and metrics

4.1.8 Process Improvement*
- 4.1.8.1 Process and product quality
- 4.1.8.2 Process analysis and modeling
- 4.1.8.3 Process measurement
- 4.1.8.4 The SEI process capability maturity model
- 4.1.8.5 Process classification

4.1.9 Software Requirements
- 4.1.9.1 Functional and non-functional requirements
- 4.1.9.2 User requirements
- 4.1.9.3 System requirements
- 4.1.9.4 The software requirements document

4.1.10 Requirements Engineering Processes*
- 4.1.10.1 Feasibility studies
- 4.1.10.2 Requirements elicitation and analysis
- 4.1.10.3 Requirements validation
- 4.1.10.4 Requirements management

4.1.11 System Models
- 4.1.11.1 Context models
- 4.1.11.2 Behavioral models
- 4.1.11.3 Data models
- 4.1.11.4 Object models
- 4.1.11.5 CASE workbenches

4.1.12 Software Prototyping*
- 4.1.12.1 Prototyping in the software process
- 4.1.12.2 Rapid prototyping techniques
- 4.1.12.3 User interface prototyping

4.1.13 Architectural Design*
- 4.1.13.1 System structuring
- 4.1.13.2 Control models
- 4.1.13.3 Modular decomposition
- 4.1.13.4 Domain-specific architectures

4.1.14 Object-oriented Design*
- 4.1.14.1 Objects and object classes
- 4.1.14.2 An object-oriented design process
- 4.1.14.3 Design evolution

4.1.15 Design with Reuse*

4.1.15.1 Component-based development
4.1.15.2 Application families
4.1.15.3 Design patterns

4.1.16 User Interface Design*

4.1.16.1 User interface design principles
4.1.16.2 User interaction
4.1.16.3 Information presentation
4.1.16.4 User support
4.1.16.5 Interface evaluation

5.0 Teaching Methodology Information

5.1 Methods:

Most of the course materials are presented by lectures. Student's participation in class discussions are simulated by asking students opinion on the specific points of the topics.

5.2 Student role:

The student will attend lectures and demonstration, participate in discussion on assigned readings, and complete required examinations. Students form team of 3-4 members to go through most of the phases of developing a project. At the end of each phase team is required to deliver a formal document that presents the product developed in that phase.

6.0 Evaluation Information

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

The student project is evaluated primarily based on the team performance. Each team is required to submit team and individual log of activities for individual evaluation consideration. Graduate students are required to write 2 three pages reports on the topic assigned by the instructor.

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

Component	Weight
Quizzes	12%
Assignments and project	30%
Mid-term examination	24%
Class participation	5%
Final examination	29%
Graduate student paper	10%

Basis for determining the final grade (course requirements and grading standards) specifying distinction between undergraduate and graduate, if applicable.

6.3 Grading type:

Percent	***Grade***	***Percent***	***Grade***
97 – 100	*A+*	*77 – 79*	*C+*
94 – 96	*A*	*70 – 76*	*C*
90 – 93	*A–*	*70 – 73*	*C–*
87 – 89	*B+*	*67 – 69*	*D+*
84 – 86	*B*	*64 – 66*	*D*
80 – 83	*B–*	*60 – 63*	*D–*
		0—59	*F*

7.0 Resource Material Information

7.1 Textbooks and/or other required readings used in course:

Ian Sommerville. Software Engineering, 9th Ed., Addison-Wesley, 2009.

7.2 Other student suggested reading materials:

F.P. Brooks. The Mythical Man-Month: Essays on Software Engineering, 2nd Ed., Addison-Wesley, 1995.

A. Fox and D. Patterson. Engineering Software as a Service: An Agile Approach Using Cloud Computing, 2nd Ed., Strawberry Canyon, 2014. http://beta.saasbook.info/

R. Pressman. Software Engineering – A Practitioner's Approach, 7th Ed., McGraw-Hill, 2009.

S. Pfleeger and J. Atlee. Software Engineering, Theory and Practice, 4th Ed., Prentice Hall, 2009.

S. Schach. Object-Oriented and Classical Software Engineering, 8th Ed., McGraw-Hill, 2010.

7.3 Current bibliography and other resources:

The textbook has a companion website that provides a large number of examples, pointers to useful sites etc. Its URL is http://www.software-engin.com/. The text web site has an excellent collection of current bibliography of resources.

The following gives a representative set of readings covering all main areas of software engineering:

7.3.1 B. Boehm. "Get Ready for Agile Methods, with Care." IEEE Computer, 35 (4), January 2002. A good discussion of the pros and cons of agile methods such as extreme programming by a leading software engineering practitioner and researcher.

7.3.2 F. Brooks. "No Silver Bullet – Essence and Accidents of Software Engineering" IEEE Computer 20.4 (1987): 10-19. Classic discussion on real-world issues facing software developers.

7.3.3 M. Conway. "How do Committees Invent?" Datamation 14.4 (1968): 28–31. Discusses the problem of working in teams and the source of 'Conway's Law'.

7.3.4 M. Lindvall and I. Rus (eds) Special Issue on Software Process Diversity. IEEE Software, 17 (4), July 2000. This special issue includes a number of interesting and useful articles on different types of process for developing software. It also includes articles covering process maturity and the CMM.

7.3.5 R. C. Martin. “Extreme programming - Development through Dialog.” IEEE Software, July 2000. A short introduction that addresses common concerns about extreme programming.

7.3.6 N. J. Nunes and J. F. Cunha. “Wisdom: A Software Engineering Method for Small Software Development Companies.” IEEE Software, 17 (4), September 2000. A discussion of a 'lightweight' method that brings some elements of systematic software engineering to small organizations who cannot afford complex methods.

7.3.7 B. Nuseibeh. “Weaving Together Requirements and Architectures.” IEEE Computer, March 2001. A short discussion of how requirements engineering and architectural design can be integrated in an evolutionary development process.

7.3.8 G. Pour, M. L. Griss and M. Lutz. “The Push to Make Software Engineering Respectable.” IEEE Computer, 33 (5), May 2000. Describes the essentials of professionalism for software engineers.

7.3.9 E. Raymond. “The Cathedral and the Bazaar” Linux Kongress (1997) June 16, 2011 <http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar>. A blueprint for today’s open source development processes.

7.3.10 M. L. Russ and J. D. McGregor. “A Software Development Process for Small Projects.” IEEE Software, 17 (4), September 2000. Suggests a software process that has been adapted for small companies.

7.3.11 R. H. Thayer. “Software System Engineering: A Tutorial.” IEEE Computer, 35 (4), April 2002. There are few articles published on this important topic. This is a good overview and introduction.

8.0 Other Information:

8.1 Accommodations statement:

8.2 Other:

8.3 Author(s):

Harvey Siy

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

CSAB Category	*Core*	*Advanced*
Data structures		
Computer organization and architecture		
Algorithms and software design		38
Concepts of programming languages		4

10.0 Oral and Written Communications:

Every student is required to submit at least 2 written reports (not including exams, tests, quizzes, or commented programs) to typically 6 pages and to make 1 oral presentations of typically 15 minutes duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

11.0 Social and Ethical Issues:

30 minutes discussions on professional and ethical responsibility in software engineering.

12.0 Theoretical content:
Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

		Contact Hours
12.1	System and software modeling and engineering process	6.0
12.2	Foundation of system validation and formal methods in software engineering	2.0
12.3	System architecture and design	1.5
12.4	Real-time and critical system concepts in software engineering	3.0
12.5	Pattern and application frameworks	1.5
12.6	Cognitive issues in human computer interaction	1.0
12.7	Cognitive issues of people management and team dynamics	1.5
12.8	Principles of cost estimation, planning and scheduling	4.5

13.0 Problem analysis:
Please describe the analysis experiences common to all course sections.

Students learn how to understand the scope of the problem, the domain that encompases the problem, analyze the requirements and constraints. The system and engineering analysis mind set provide necessary skills that enable students to study the subject materials in different sections as a part of whole “system” with an “engineering” mind-set.

14.0 Solution design:
Please describe the design experiences common to all course sections.

Students learn to make and implement a variety of high and low-level design decisions, including: system and software architecture, subsystem, component, and interface design. Following the “engineering” mind set design enables students to develop solutions based on the analysis of the problem.

CHANGE HISTORY

Date	***Change***	***By whom***	***Comments***
09/25/2002	Initial ABET version	Zand	
06/13/2003	Cleanup	Wileman	
11/19/2008	Course textbooks updated	Siy	
11/20/2008	Update of course objectives	Siy	

11/20/2008	Insertion of table mapping course objectives to program outcomes	Siy	
6/16/2011	Updated bibliography	Siy	
7/29/2011	Updated format to standard template	Siy	
3/3/2014	Updated bibliography	Siy	

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

Department and Course Number	CSCI 4830
Course Title	Introduction to Software Engineering
Course Coordinator	Harvey Siy
Total Credits	3
Date of Last Revision	November 20, 2008

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. Perform analysis and design of small and medium-sized software project using structured methods.	**S**	**S**	**S**			**S**	**X**			**S**	**S**
2. Be able to participate in design of small and medium-sized software project using object-oriented software development methodologies.	**S**	**S**	**S**					**X**	**S**	**S**	**S**
3. Prepare software project management documents.	**S**			**S**		**S**		**X**	**S**		
4. Be able to participate in a project team.				**S**		**X**					
5. Develop parts/whole prototype as well as implementation of small or medium-sized software projects.		**S**	**S**	**S**				**X**	**S**	**S**	**S**
6. Introduce socio-technical and ethical issues in the development of real-world software systems.					**S**		**S**	**X**			
7.											