The University of Georgia

Department of Computer Science

Course Information Sheet

CSCI 4050

Software Engineering

Brief Course Description
(50-words or less)

Full cycle of a software system development effort, including requirements definition, system analysis, design, implementation, and testing. Special emphasis is placed on system analysis and design. The design phase includes development of a user interface. A large term project incorporates the full software life cycle.

Extended Course Description / Comments

In this course, the students learn the principles of Software Engineering. Although several of the major software design techniques are discussed, the course concentrates on Object-Oriented Design (OOD). The course begins with a discussion of the software development process and what constitutes well-engineered software. The next subject is the requirements elicitation and requirements specification. The students learn how to structure and define functional and non-functional requirements. The next part of the course is devoted to requirements analysis, where several UML diagrams are introduced to represent a variety of object-oriented models. This phase is followed by system design, which includes software architecture specification as well as an introduction to design patterns. The construction phase covers a number of implementation techniques, including mapping models to code. Finally, the students learn a variety of software verification and testing techniques. A large portion of the course is devoted to implementation techniques suitable for the creation of reliable and maintainable software. The course involves a large team-based software project, which is developed during the entire semester. The students learn the principles of project management and team software design and development, as well.

This course is part of the BS-CS Teamwork Requirement; students In CSCI 4050 are required to work in teams of size greater than 2.

Pre-Requisites and/or Co-Requisites

CSCI 2720
Data Structures

Approved Textbooks
(if more than one listed, the textbook used is up to the instructor's discretion)

Author(s): Bernd Bruegge and Allen H. Dutoit
Title: Object-Oriented Software Engineering. Using UML, Patterns, and Java, Prentice Hall, 2010.
Edition: 3-rd edition
ISBN-13: 978-0136061250

Specific Learning Outcomes (Performance Indicators)

These are a (non-exhaustive) list of specific, measurable outcomes, as they relate to course and program objectives.

This course presents a survey of topics in software engineering most relevant to students studying computer science. At the end of the semester, all students will be able to do the following:

1. Identify and differentiate phases of a typical software process and how it relates to the software life cycle.
2. Create functional requirement specifications in the form of use cases and differentiate between functional and non-functional requirements.
3. Create UML class diagrams representing a domain object model.
4. Create UML sequence, state, and communication diagrams in order to analyze and model use cases.
5. Create a software architecture specification, including subsystem decomposition and subsystem interface descriptions.
6. Create and test an implementation of a software system based on the

previously created models.

7. Deliver a presentation and demonstration of a functioning software system and of the results of its testing.
8. Communicate and effectively function as a member of a software development team.

Relationship Between Student Outcomes and Learning Outcomes

		Student Outcomes										
		a	b	c	d	e	f	g	h	i	j	k
Learning Outcomes	1			•						•		•
	2	•	•	•	•					•	•	•
	3	•	•	•	•					•	•	•
	4	•	•	•	•					•	•	•
	5	•	•	•	•					•	•	•
	6	•	•	•	•					•	•	•
	7			•	•							
	8				•							

Student Outcomes

(These are ABET-specified and should not be changed)

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 practice.
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.

Major Topics Covered
(Approximate Course Hours)

3 credit hours = 37.5 contact hours
4 credit hours = 50 contact hours

Note: Exams count as a major topic covered

Software Engineering and Software Process (3-hours)
Team and project management (2-hours)
Requirements elicitation and specification (4-hours)
Use case modeling (3-hours)
Requirements analysis (4-hours)
UML diagrams (4-hours)
Static and dynamic modeling (3-hours)
System design and architectural styles (4-hours)
Design patterns (3-hour)
Detailed (object) design (4-hours)
Object Constraint Language (2-hours)
Implementation techniques (5-hours)
Source code management (2-hours)
Persistence and storage systems (2-hours)
Verification and Testing (3-hours)
Software demonstration (2-hours)

Assessment Plan for this Course

Each time this course is offered, the class is initially informed of the Course Outcomes listed in this document, and they are included in the syllabus. At the end of the semester, an anonymous survey is administered to the class where each student is asked to rate how well the outcome was achieved. The choices provided use a 5-point Likert scale containing the following options: Strongly agree, Agree, Neither agree or disagree, disagree, and strongly disagree. The results of the anonymous survey are tabulated and results returned to the instructor of the course.

The course instructor takes the results of the survey, combined with sample student responses to homework and final exam questions corresponding to course outcomes, and reports these results to the ABET committee. If necessary, the instructor also writes a recommendation to the ABET committee for better achieving the course outcomes the next time the course is offered.

How Data is Used to Assess Program Outcomes

Each course Learning Outcome, listed above, directly supports one or more of the Student Outcomes, as is listed in "Relationships between Learning Outcomes and Student Outcomes". For CSCI 4050, Student Outcomes (a) (b) (c) (d) (i) (j) and (k) are supported.

Course Master

Dr. Krzysztof Kochut

Course History

08/2008	Course Approved in CAPA
02/2012	Course Information Sheet Created
12/7/2012	Course added to BS-CS Teamwork Requirement block