Department of Computer Science
Franklin College of Arts and Sciences
UNIVERSITY OF GEORGIA

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.

Pre-Requisites and/or Co-Requisites

Prerequisite: CSCI 2720 (Data Structures) OR CSCI 2725 (Data Structures for Data Science)

Required, Elective or Selected Elective

Required Course

Approved Textbook

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)

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 and the different software process models.
2. Create functional requirement specifications in the form of use cases and user stories and differentiate between functional and non-functional requirements.

3. Develop static and dynamic UML diagrams to model both the structural and behavioral aspects of the software system throughout the different phases of the development life cycle.
4. Create a software architecture specification, including subsystem decomposition and subsystem interface descriptions.
5. Communicate and effectively function as a member of a software development team to develop a software system based on its specification and previously created models.
6. As a team, deliver a coherent and professional presentation and demonstration of a functioning software system and the results of its testing.

ABET Learning Outcomes

A. Graduates of the program will have an ability to: Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions.
B. Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.
C. Communicate effectively in a variety of professional contexts.
D. Recognize professional responsibilities and make informed judgments in computing practice based on legal and ethical principles.
E. Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline.
F. Apply computer science theory and software development fundamentals to produce computing-based solutions.

Relationship Between Student Outcomes and Learning Outcomes

	ABET Learning Outcomes						
Specific Learning Outcomes		a	b	c	d	e	f
	1	●					●
	2	●		●		●	●
	3	●	●	●		●	●
	4	●	●	●			●
	5		●			●	●
	6		●	●		●	●

Major Topics Covered

1. Software Engineering and Software Process (3-hours)
2. Team and project management (2-hours)
3. Requirements elicitation and specification (4-hours)
4. Use case modeling (3-hours)
5. Requirements analysis (4-hours)
6. UML diagrams (4-hours)
7. Static and dynamic modeling (3-hours)
8. System design and architectural styles (4-hours)
9. Design patterns (3-hour)
10. Detailed (object) design (4-hours)
11. Object Constraint Language (1-hours)
12. Implementation techniques (5-hours)

13. Source code management (2-hours)
14. Persistence and storage systems (2-hours)
15. Verification and Testing (3-hours)
16. Software demonstration (3-hours)

Course Master	Dr. Krzysztof Kochut
Modified	7/15/2020 by Dr. Krzysztof Kochut and Dr. Eman Saleh
Approved	No