UNIVERSITY OF CAPE COAST
COLLEGE OF AGRICULTURE AND NATURAL SCIENCES
SCHOOL OF PHYSICAL SCIENCES
DEPARTMENT OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY

COURSE INFORMATION
Course Code: INF307
Course Title: Software Engineering
Credit Hours: 3
Semester: Two
Class Hours: Mondays 2:30 pm-4:30 pm @ SW15

CONTACT DETAILS OF INSTRUCTOR
Name: Elliot Attipoe
Email: eattipoe@ucc.edu.gh
Contact: 024 387 9458
Office Hours: Wednesday – Friday, 10:00 am – 2:00 pm

RREADING TEXT(S)
1. Sommerville, I. (2017), Software Engineering, 10th Ed, Addison-Wesley, USA.
2. Sommerville, I. (2019), Engineering Software Products: An Introduction to Modern Software Engineering, USA.
3. Pressman, R. S. & Maxim, B. (2019). *Software engineering: a practitioner's approach, 9th Ed,* McGraw Hill, USA
4. Kendall, K. E. and Kendall, J. E. (2019) Systems Analysis and Design, 10th edition, Hoboken: Pearson
5. Sethi, R. (2022). Software Engineering: Basic Principles and Best Practices. Cambridge University Press.

Course Description
This course is designed to provide students with a comprehensive understanding of software development principles, methodologies, and practices. The course equips students with the necessary knowledge and skills to design, develop, test, and maintain high-quality software solutions that meet real-world requirements. Through theoretical knowledge, hands-on coding projects, and collaborative activities, students will be prepared to excel in the dynamic field of software engineering and contribute to creating innovative software applications. The course will include a programming project in which teams of 4-6 students take a project from requirements through implementation. The approaches that would be used in the delivery of this course would prepare students to be mindful of gender roles and issues about equity and inclusion by paying attention to the learning progress of all students.

Course Objectives

At the end of the course, the students will be able to:

i. Describe and compare various software development methods and understand the context in which each approach might be applicable.
ii. Familiar with mainstream languages used to model and analyze object designs (e.g., UML).
iii. Explain the scope of the software maintenance problem and demonstrate the use of several tools for reverse engineering software.
iv. evaluate the effectiveness of an organization's software development practices, suggest improvements, and define a process improvement strategy.
v. use state-of-the-art tools and techniques for large-scale software systems development.
vi. Implement the major software development methods in practical projects.

COURSE CONTENT

- Introduction to Software Engineering
  - Professional software development
  - Software engineering ethics
- Software Development Process
  - Software process models
  - Process activities
  - Coping with change
  - The rational unified process
- Agile Software Development
  - Agile methods
  - Plan-driven and agile development
  - Extreme programming
  - Agile project management
- Requirement Engineering
  - Functional and non-functional requirements
  - The software requirements document
  - Requirement specification
  - Requirement engineering processes
- System Modeling
  - Use cases
  - Sequence diagram
  - Class diagram
  - Activity diagram
- Design and implementation
  - Object-oriented design using the UML
  - Design patterns

    - Implementation issues
    - Open source development
- Software testing and evolution
    - Development testing
    - Test-driven development
    - User testing
    - Evolution processes
    - Software maintenance

TEACHING AND LEARNING STRATEGIES

- case studies
- simulations
- discussion method
- group work
- project
- Problem-Solving
- lecture and exposition
- hands-on/lab sessions
- oral presentations
- role playing

ASSESSMENT

A combination of formative and summative assessments, including group tasks, quizzes, assignments, and examinations, will be used.

Student Assessment System

CONTINUOUS ASSESSMENT: 40%
END-OF-SEMESTER EXAMINATION: 60%

The Continuous Assessment component includes take-home assignments, class quizzes and tests, and project work. This component allows students to demonstrate their abilities on a wider variety of learning tasks and work environments than possible under formal examination conditions. For example, through Continuous Assessment, students can learn the values and processes of teamwork plan and solve real-life problems.

GRADING SCALE

The University uses letter grades and numerical weightings corresponding to the letter grades. The numerical weightings reflect the quality of performance. Total numerical weightings reflect the quality of performance. Total raw scores (combination of continuous assessment and end-of-semester examination) are converted according to the following scheme:

RAW SCORE	GRADE	CREDIT VALUE	INTERPRETATION
80 – 100	A	4.0	Excellent
75 – 79	B+	3.5	Very Good
70 – 74	B	3.0	Good
65 – 69	C+	2.5	Average
60 – 64	C	2.0	Fair
55 -59	D+	1.5	Barely Satisfactory
50 – 54	D	1.0	Weak Pass
Below 50	E		Fail

COURSE POLICY

- **Attendance**: 100% attendance is required. Attendance to all sessions is compulsory. By the University regulations you must support any absenteeism by the appropriate report.

- **Code of conduct**: You should be punctual at all sessions and conduct yourself at same in a professional manner. You must also switch your phone to silent mode.

- **Cheating/Plagiarism**: Cheating and plagiarism of any kind will not be tolerated and will attract the appropriate sanctions as stipulated in the students' handbook.

- **Assignment submission**: All work should be submitted by the deadline.
- **Referencing:** Use APA style in your essays/project write-up as stipulated in the School of Graduate Studies theses and dissertation guide.

SCHEDULE

Week	Content	Objective *Students will be able to:*	Activities
1	Introduction to Software Engineering	i. Understand what software engineering is and why it is important; ii. understand the development of different types of software systems may require different software engineering techniques; iii. understand some ethical and professional issues that are important for software engineers	Lecture/Discussion

2-3	Proposal presentation	i. Analyze a problem and develop an initial solution. ii. Apply a multi-disciplinary approach to designing a project. iii. Demonstrate the ability to work independently and in a team. iv. Demonstrate the ability to communicate effectively.	Discussion/group work/ presentations
4	Software Development Process	i. Understand the concepts of software processes and software process models; ii. have been introduced to three generic software process models and when they might be used; know about the fundamental process activities of software requirements engineering, software development, testing, and evolution; iii. understand why processes should be organized to cope with changes in the software requirements and design; iv. understand how the Rational Unified Process integrates good software engineering practice to create adaptable software processes.	Group work/ Class Discussion/Case Studies
5	Agile Software Development	i. Understand the rationale for agile software development methods, the agile manifesto, and the differences between agile and plan-driven development; ii. know the key practices in extreme programming	Project/Discussion/ Case Studies/Lecture

		and how these relate to the general principles of agile methods; iii. understand the Scrum approach to agile project management; iv. be aware of the issues and problems of scaling agile development methods to develop large software systems.	
6	Requirement Engineering	i. Understand the concepts of user and system requirements and why these requirements should be written in different ways; ii. understand the differences between functional and nonfunctional software requirements; iii. understand how requirements may be organized in a software requirements document; iv. understand the principal requirements of engineering activities of elicitation, analysis, and validation, and the relationships between these activities; v. understand why requirements management is necessary and how it supports other requirements engineering activities.	Project/ Simulation, Problem-Solving, Discussions, Hands-on/lab sessions
7-8	System Modeling	i. Understand how graphical models can be used to represent software systems; ii. understand why different types of models are required and the	Project/ Simulation, Problem-Solving, Discussions, Hands-on/lab sessions

		fundamental system modeling perspectives of context, interaction, structure, and behavior; iii. have been introduced to some of the diagram types in the Unified Modeling Language (UML) and how these diagrams may be used in system modeling;	
9-10	Group Presentation	i. Apply core knowledge areas of software engineering to implement a project. ii. Use modern tools and technologies to implement a project. iii. Demonstrate the ability to work independently and in a team. iv. Demonstrate the ability to communicate effectively. v. Commit to professional, ethical, legal, security and social issues and responsibilities throughout a project. vi. Produce a complete report for a project.	Discussion/group work/ presentations
11	Design and Implementation	i. understand the most important activities in a general, object-oriented design process; ii. understand some of the different models that may be used to document an object-oriented design; iii. know about the idea of design patterns and how these are a way of reusing design knowledge and	Group work and presentations

		experience; iv. have been introduced to key issues to consider when implementing software, including software reuse and open-source development.	
12	Software testing and evolution	i. understand the stages of testing from testing during development to acceptance testing by system customers; ii. have been introduced to techniques that help you choose test cases that are geared to discovering program defects; iii. understand test-first development, where you design tests before writing code and run these tests automatically; iv. know the important differences between component, system, and release testing and be aware of user testing processes and techniques. v. understand that change is inevitable if software systems are to remain useful and that software development and evolution may be integrated in a spiral model; vi. understand software evolution processes and influences on these processes; vii. have learned about different types of software maintenance and the factors that affect maintenance costs; and viii. understand how legacy	Group work and presentations

		systems can be assessed to decide whether they should be scrapped, maintained, reengineered, or replaced.	
13	Revision		