BIRZEIT UNIVERSITY

Computer Science Department

Software Engineering (COMP 433)
Fall 2021/2022

Instructors:

- Dr. Samer Zein – Section 1 and 3 – szain@birzeit.edu
- Mr. Saad Mansour - Section 2 and 4 - smansour@birzeit.edu

A. **Text Book:** Bruegge and Dutoit, ***Object-Oriented Software Engineering Using UML, Patternd, and Java***, **3rd** Edition, Prentice Hall, 2013.

Suggested reading:

B. Sommerville I. (2001,2004-2010) ***Software Engineering***, 9th Edition, Addison-Wesley, Harlow, Essex, UK (older editions can also be suitable for this course)
C. Stevens P. with Pooley, R. (2005) ***Using UML: Software Engineering with Objects and Components***, 2nd Edition, Addison-Wesley, Harlow, Essex, UK
D. Jeffrey A. Hoffer, Joey F. George, Joseph S. Valacich. (2005) ***Modern System Analysis and Design*** 4th - 7th Edition, (2013), Prentice Hall.
E. Roger Pressman, ***Software Engineering: A Practitioner's Approach***, 7-8th edition, McGraw-Hill; 2014.
F. L.A. Maciaszek, ***Requirements Analysis and System Design: Developing Information Systems with UML***, 1-3rd Edition, Addison Wesley, 2007.

Introduction:
Software engineering is the discipline concerned with the application of theory, knowledge, and practice for effectively and efficiently building software systems that satisfy the requirements of users and customers. Software engineering is applicable to small, medium, and large-scale systems. It encompasses all phases of the life cycle of a software system.
Software engineering employs engineering methods, processes, techniques, and measurement. It benefits from the use of tools for managing software development; analyzing and modeling software artifacts; assessing and controlling quality; and for ensuring a disciplined, controlled approach to software evolution and reuse. Software development, which often involves a team of developers, requires choosing the suitable tools, methods, and approaches that are most applicable for a given development environment.
The elements of software engineering are applicable to the development of software in any computing application domain where professionalism, quality, schedule, and cost are important in producing a software system.

Aims:
To provide an overall understanding of the fundamental concepts, and practical methods for engineering software. It will provide an in-depth understanding of software engineering approaches centred around and grounded into practical context. The course will equip students with foundational knowledge and practical skills to apply software engineering methods and techniques for engineering reasonably large software systems. It will also provide students with analytical means for assessing and evaluating factors that influence the selection and use of appropriate software engineering methods to appreciate their practical applications and their limitations.

Learning Outcomes
Students successfully completing this course will:

Development and Understanding

Have a good understanding of software engineering foundational concepts
Have a good knowledge of software engineering techniques and methods
Be able to demonstrate critical awareness of personal responsibility and professional codes of conduct in the context of software systems development

Cognitive/Intellectual Skills

Have the ability to analyse complex areas of knowledge regarding software engineering
Know how to critically evaluate software engineering approaches and methods
Be able to assess and analyse considerations of contextual and circumstantial situations when/where to apply and use software engineering methods and models
Be able to apply analysis and design considerations to the design of software systems
Have the ability to recognise and appreciate software engineering technical limitations

Key/transferable Skills

Be able to manage their own learning and information in support of their study
Know how to undertake self-evaluation of their work
Be able to work effectively within a team and support the various team functions
Be able to document and present their own learning and work outputs in suitable written formats.

Practical Skills

Be able to apply software engineering modelling methods (UML) and software system tools
Be able to practice the stages of software development life cycle through a team project.
Be able to practice applying software engineering modelling methods and approaches through a team project.
Be able to practice the presentation of their own learning and work outputs.

Syllabus:

Introduction and general background to Software Engineering	1	Chapter 1 (A)/ (B) (+ other introductory material)
Software processes, models and techniques (Software Development Life Cycle)	2	Chapter 2 (B) [+Chapter 3 (B)]
Software Requirements Engineering/Elicitation	3	Chapter 4 (A) / (B)
System Modelling and Analysis Using Unified modelling Language (UML)	4	Chapter 5 (B) [Chapter 2 (A)+Chapter 5 (A)] (+ UML 2.0 from UML standard (C))
System Design Detailed Design	3 2	Chapter 6 (A)+Chapter 7 (B) [Chapter 6 (B) + Chapter 7.1 (B)]

Formative Assessment:

Mid Term Exam and Quizzes 25%
Group Project and Assignments 40%

Summative assessment:

Final Exam 35%

* Rational Rose UML tool is available, for students with an educational license, to develop UML diagrams. Using any other UML authoring tool is acceptable.

Notes:

- The group project is a substantial and essential part of the course and carries significant part of the course assessment. It requires you to form a group of 3-4 of your peers in the first two weeks of the course.
- The project has three phases to complete and each requires submission in a written format. Thus timely project submissions are accepted only.
- Professional code of conduct are part of the aims of this course, thus strive for a professional presentation of your project submissions.