SYLLABUS

Software engineering

University year 2025-2026

1. Information regarding the programme

1.1. Higher education institution	Babeş-Bolyai University
1.2. Faculty	Faculty of Mathematics and Computer Science
1.3. Department	Department of Computer Science
1.4. Field of study	Computers and information technology
1.5. Study cycle	Bachelor
1.6. Study programme/Qualification	Information engineering
1.7. Form of education	Full time

2. Information regarding the discipline

2.1. Name of the discipline	**Software engineering**					Discipline code	**MLE5177**
2.2. Course coordinator					Assoc. Prof. Vescan Andreea, PhD		
2.3. Seminar coordinator					Assoc. Prof. Vescan Andreea, PhD		
2.4. Year of study	3	2.5. Semester	6	2.6. Type of evaluation	E	2.7. Discipline regime	DD

3. Total estimated time (hours/semester of didactic activities)

3.1. Hours per week	**5**	of which: 3.2 course	**2**	3.3 seminar/laboratory/project	**1 S, 1 LP, 1P**
3.4. Total hours in the curriculum	70	of which: 3.5 course	28	3.6 seminar/laboratory/project	**42**
Time allotment for individual study (ID) and self-study activities (SA)					**hours**
Learning using manual, course support, bibliography, course notes (SA)					20
Additional documentation (in libraries, on electronic platforms, field documentation)					20
Preparation for seminars/labs, homework, papers, portfolios and essays					20
Tutorship					10
Evaluations					10
Other activities:					
3.7. Total individual study hours	**80**				
3.8. Total hours per semester	**150**				
3.9. Number of ECTS credits	**6**				

4. Prerequisites (if necessary)

4.1. curriculum	• Fundamentals of programming • Object-Oriented programming
4.2. competencies	• Programming in a high-level object-orietend language

5. Conditions (if necessary)

5.1. for the course	• Videoprojector
5.2. for the seminar /lab activities	• Computers • UML Case Tool • Java/.NET IDE

6.1. Specific competencies acquired [1]

[1] One can choose either competences or learning outcomes, or both. If only one option is chosen, the row related to the other option will be deleted, and the kept one will be numbered 6.

Professional/essential competencies	• Problem solving using specific computer science and computer engineering tools • Design and integration of information systems using technologies and programming environments
Transversal competencies	• Honorable, responsible, ethical behavior, in the spirit of the law, to ensure the professional reputation • Identifying, describing and conducting processes in the project management field, undertaking different team roles and clearly and concisely describing own profesional results, verbally or in writing • Demonstrating initiative and pro-active behavior for updating professional, economical and organizational culture knowledge

6.2. Learning outcomes

Knowledge	The student knows: • The graduate knows and understands the basic concepts, theories and methods of Computer and Information Technology and is able to use them appropriately in professional communication. • The graduate has the ability to choose and use programming paradigms (procedural, object-oriented, functional) to create software applications appropriate to the specific field of the developed application. • The graduate has the necessary skills to apply different methods and tools for analyzing and visualizing research results. The graduate is able to write a scientific report. • The graduate has the necessary knowledge related to the stages of the software life cycle and software process models. • The graduate knows the concepts related to software modeling and can implement functional and non-functional requirements described in specific documents for the analysis and design of software systems.
Skills	The student is able to • The graduate is able to design / implement hardware, software and communications components using design methods, languages, algorithms, data structures, protocols and technologies, and evaluate their functional and non-functional characteristics based on metrics. • The graduate is able to develop systems and applications for the maintenance and use of hardware, software and communications systems. • The graduate performs the testing and qualitative evaluation of the functional and non-functional characteristics of the information systems, based on specific criteria. • The graduate has the ability to develop, design and create new applications, systems or products using best practices in the field of computer science.
Responsibility and autonomy:	The student has the ability to work independently to obtain • The graduate is familiar with tools used for testing, debugging, validating software applications.

7. Objectives of the discipline (outcome of the acquired competencies)

7.1 General objective of the discipline	• Aquiring knowledge of and applying sound concepts, principles and engineering techniques when building software systems

7.2 Specific objective of the discipline	• Aquiring knowledge of software lifecycle stages and process models • Understanding software modeling • Aquiring knowledge of and applying model-based software development techniques • Getting used to correctly apply the UML language • Aquiring ability to use UML Case tools • Aquiring basic project management knowledge • Aquiring knowledge of software development methodologies, both traditional and agile

8. Content

8.1 Course	Teaching methods	Remarks
1. Introduction to Software Engineering: motivation, definitions, concepts, activities	Explanation, conversation, discussing case studies	
2. Software lifecycle stages. Software process models	Explanation, conversation, discussing case studies	
3. Software complexity management techniques (abstraction, decomposition, modeling). Modeling in Software Engineering: definitions, model types and modeling tools	Explanation, conversation, discussing case studies	
4. Introduction to the UML language: concepts, diagram types, syntax/semantics, tools	Explanation, conversation, discussing case studies	
5. Requirements Elicitation: concepts, activities, examples	Explanation, conversation, discussing case studies	
6. Requiements Analysis: concepts, activities, examples	Explanation, conversation, discussing case studies	
7. System Design: concepts, principles, activities	Explanation, conversation, discussing case studies	
8. Object Design: concepts, principles, activities	Explanation, conversation, discussing case studies	
9. Object Design - Design Patterns	Explanation, conversation, discussing case studies	
10. Object Design – Interface Specification. Design by Contract – using assertions in modeling	Explanation, conversation, discussing case studies	
11. System Implementation. Model-based code generation: concepts, principles, activities, examples	Explanation, conversation, discussing case studies	
12. Software Verification and Validation	Explanation, conversation, discussing case studies	
13. Software Management: concepts and activities	Explanation, conversation, discussing case studies	
Bibliography [1] Booch, G., Rumbaugh, J., Jacobson, I., *The Unified Modeling Language User Guide - V.2.0*, Addison Wesley, 2005.		

[2] Bruegge, B., Dutoit, A., *Object-Oriented Software Engineering Using UML, Patterns and Java - 3rd Edition,* Prentice Hall, 2009.
[3] Fowler, M. et al., *Refactoring - Improving the Design of Existing Code*, Addison Wesley, 1999.
[4] Fowler, M., Scott, K., *UML Distilled: A Brief Guide to the Standard Object Modeling Language -2nd ed*., Addison-Wesley, 1999.
[5] Gamma, E., Helm, R., Johnson, R., Vlissides, J., *Design Patterns*, Addison-Wesley, 1996.
[6] Martin, R.C., *Agile Software Development: Principles, Patterns, and Practices*, Prentice Hall, 2002.
[7] Pârv, B., *Analiza si proiectarea sistemelor*, Univ. Babeș-Bolyai, CFCID, Facultatea de Matematică și Informatică, Cluj-Napoca, 2004.
[8] Pressman, R.S., *Software Engineering - A Practitioners Approach - 6th ed.*, McGraw-Hill, 2005.
[9] Schach, S.R., *Object-Oriented and Classical Software Engineering - 6th ed.*, McGraw-Hill, 2005.
[10] Sommerville, I., *Software Engineering - 8th edition*, Addison-Wesley, 2006.

8.2 Seminar	Teaching methods	Remarks
1. Using Use Case Diagrams to describe a functional model: concepts, relations, syntax, use case description templates	explanation, conversation, arguing, exemplifying	
2. Using Class Diagrams to describe structural models: concepts, relations, syntax, problem domain model vs. solution model	explanation, conversation, arguing, exemplifying	
3. Using Sequence/Communication Diagrams to describe dynamic models: concepts. syntax, equivalence	explanation, conversation, arguing, exemplifying	
4. Using Statechart Diagrams to describe dynamic models. The State Design Pattern	explanation, conversation, arguing, exemplifying	
5. The use of assertions in modeling. Design by Contract	explanation, conversation, arguing, exemplifying	
6. Automatic code generation based on UML/OCL models	explanation, conversation, arguing, exemplifying	
7. Testing: concepts, principles, tools	explanation, conversation, arguing, exemplifying	
8.3 Laboratory		
1. Agile methodologies: planning software development. Investigating various UML/OCL Case Tools (ex. StarUML, OCLE)	explanation, conversation, arguing, exemplifying	
2. Using an UML Case Tool for drawing Use Case Diagrams	explanation, conversation, arguing, exemplifying	
3. Using an UML Case Tool for drawing Class Diagrams corresponding to the problem domain	explanation, conversation, arguing, exemplifying	
4. Using an UML Case Tool for drawing Sequence/Communication Diagrams and refining the structural model	explanation, conversation, arguing, exemplifying	
5. Using an UML Case Tool for drawing Statechart Diagrams	explanation, conversation, arguing, exemplifying	
6. Using an UML/OCL Case Tool for specifying/evaluating assertions on UML models	explanation, conversation, arguing, exemplifying	
7. Using an UML/OCL Case Tool for code generation	explanation, conversation, arguing, exemplifying	

8.4 Project		
1. Agile methodologies: planning software development. Investigating various UML/OCL Case Tools (ex. StarUML, OCLE)	explanation, conversation, arguing, exemplifying	
2. Using an UML Case Tool for drawing Use Case Diagrams	explanation, conversation, arguing, exemplifying	
3. Using an UML Case Tool for drawing Class Diagrams corresponding to the problem domain	explanation, conversation, arguing, exemplifying	
4. Using an UML Case Tool for drawing Sequence/Communication Diagrams and refining the structural model	explanation, conversation, arguing, exemplifying	
5. Using an UML Case Tool for drawing Statechart Diagrams	explanation, conversation, arguing, exemplifying	
6. Using an UML/OCL Case Tool for specifying/evaluating assertions on UML models	explanation, conversation, arguing, exemplifying	
7. Using an UML/OCL Case Tool for code generation	explanation, conversation, arguing, exemplifying	
Bibliography		

9. Corroborating the content of the discipline with the expectations of the epistemic community, professional associations and representative employers within the field of the program

- The course obeys to the ACM/IEEE curricula guidelines for computer science study programs
- Similar courses are taught at most universities in Romania having similar study programs
- Software companies view this course as offering important background knowledge for future software developers

10. Evaluation

Activity type	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Percentage of final grade
10.1 Course	Knowledge of the basic software engineering concepts and principles taught	Written exam	50%
	Software modeling knowledge and ability to use the UML language in this purpose		
10.1 Seminar	Software engineering concepts and principles taught	Seminar activity	10%

10.5 Laboratory/Project	Applying aquired knowledge in building a small/medium-sized software system	Project	40%
10.6 Minimum standard of performance			
• The final grade computed with the given formula must be at least 5 in order to pass the exam. • Attendance at least 5 seminars and 6 laboratories is MANDATORY for passing the discipline. Students who do not attend at least 5 seminars and 6 laboratories cannot take the exam even in the retake session.			

11. Labels ODD (Sustainable Development Goals)[2]

Not applicable.

Date: Signature of course coordinator: Assoc. Prof. Vescan Andreea, PhD

Signature of seminar coordinator: Assoc. Prof. Vescan Andreea, PhD

Date of approval:

Signature of the head of department: Assoc.prof.phd. Adrian STERCA

[2] Keep only the labels that, according to the *Procedure for applying ODD labels in the academic process*, suit the discipline and delete the others, including the general one for *Sustainable Development* – if not applicable. If no label describes the discipline, delete them all and write „*Not applicable.*".