SYLLABUS

1. Information regarding the programme

1.1 Higher education institution	**Babes Bolyai University**
1.2 Faculty	**Faculty of Mathematics and Computer Science**
1.3 Department	**Department of Computer Science**
1.4 Field of study	**Computer Science**
1.5 Study cycle	**Bachelor**
1.6 Study programme / Qualification	**Computer Science**

2. Information regarding the discipline

2.1 Name of the discipline (en) (ro)			**Software engineering** **Ingineriea sistemelor soft**				
2.2 Course coordinator			**Lect. Dr. Zsigmond Imre**				
2.3 Seminar coordinator			**Lect. Dr. Zsigmond Imre**				
2.4. Year of study	**2**	2.5 Semester	**2**	2.6. Type of evaluation	**C**	2.7 Type of discipline	**Compulsory**
2.8 Code of the discipline		MLE5011					

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

3.1 Hours per week	4	Of which: 3.2 course	2	3.3 seminar/laboratory	1S + 1L
3.4 Total hours in the curriculum	56	Of which: 3.5 course	28	3.6 seminar/laboratory	14/14
Time allotment:					hours
Learning using manual, course support, bibliography, course notes					27
Additional documentation (in libraries, on electronic platforms, field documentation)					14
Preparation for seminars/labs, homework, papers, portfolios and essays					23
Tutorship					10
Evaluations					20
Other activities:					

3.7 Total individual study hours	94
3.8 Total hours per semester	150
3.9 Number of ECTS credits	6

4. Prerequisites (if necessary)

4.1. curriculum	• Object-Oriented Programming
4.2. competencies	• Average programming skills in a high level object-oriented programming language

5. Conditions (if necessary)

5.1. for the course	• Projector
5.2. for the seminar /lab activities	• Laboratory with enough computers for students who do not have personal laptops

6. Specific competencies acquired

Professional competencies	C2.3 - Ability to work independently and in a team in order to develop software complying with industrial standards. C2.5 - Understanding the role of different artifacts used in the process of software development and acquiring the ability of realizing and using these artifacts
Transversal competencies	CT2 - Ability to create software beginning with model construction, continuing with model verification and model transformation in code, realizing and using testing models CT3 - Ability to use a software methodology to produce quality software from analyzing software requirements to code generation and software testing

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

7.1 General objective of the discipline	• Be able to understand software production life cycle • Improved skills on developing software
7.2 Specific objective of the discipline	• Be able to develop software as a team • Understand the best practices deployed in the software industry • Be able to better communicate with others on technical matters • Understand various architectures

8. Content

8.1 Course	Teaching methods	Remarks
1. Introduction to Software engineering	Exposure: description, explanation, examples, discussion of case studies	
2. Software projects	Exposure: description, explanation, examples, discussion of case studies	
3. Requirements elicitation	Exposure: description, explanation, examples, discussion of case studies	
4. Working in teams	Exposure: description, explanation, examples, discussion of case studies	
5. Desktop development	Exposure: description, explanation, examples, discussion of case studies	
6. Data storage solutions	Exposure: description, explanation, examples, discussion of case studies	
7. Code quality	Exposure: description, explanation, examples,	

	discussion of case studies	
8. Dependency management	Exposure: description, explanation, examples, discussion of case studies	
9. Web development	Exposure: description, explanation, examples, discussion of case studies	
10. Managing business logic	Exposure: description, explanation, examples, discussion of case studies	
11. Useful UML diagrams	Exposure: description, explanation, examples, discussion of case studies	
12. Software architectures	Exposure: description, explanation, examples, discussion of case studies	
13. Cloud development	Exposure: description, explanation, examples, discussion of case studies	
14. Exam		
Bibliography 1. Andrew Troelsen, Phil Japikse: Pro C# 10 with .NET 6 2. Robert C. Martin: Clean code 3. Robert C. Martin: Clean architecture 4. Roy Osherove: The art of unit testing 5. Scott Chacon: Pro Git 6. Adam Freeman: Pro ASP.NET Core 6		
8.2 Seminar /	Teaching methods	Remarks
1. Requirements gathering	Explanation, Dialogue, debate, case studies, examples, proofs	
2. Use case diagrams, class diagrams, GUI design	Explanation, Dialogue, debate, case studies, examples, proofs	
3. Desktop development	Explanation, Dialogue, debate, case studies, examples, proofs	
4. Code review + Unit testing	Explanation, Dialogue, debate, case studies, examples, proofs	
5. Web development, orm	Explanation, Dialogue, debate, case studies, examples, proofs	
6. Sequence and flow diagrams	Explanation, Dialogue, debate, case studies, examples, proofs	
7. Message bus	Explanation, Dialogue, debate, case studies, examples, proofs	
8.3 Laboratory	Explanation, Dialogue, debate, case studies, examples, proofs	
1. Environment setup and C#	Explanation, Dialogue, debate, case studies,	

	examples, proofs	
2. Reusable model development	Explanation, Dialogue, debate, case studies, examples, proofs	
3. Planning initial version of project with the use of uml and project management techniques	Explanation, Dialogue, debate, case studies, examples, proofs	
4. Software development in teams	Explanation, Dialogue, debate, case studies, examples, proofs	
5. Code review, refactoring, unit and integration testing	Explanation, Dialogue, debate, case studies, examples, proofs	
6. Web development in larger teams	Explanation, Dialogue, debate, case studies, examples, proofs	
7. Multi-platform support in even larger teams	Explanation, Dialogue, debate, case studies, examples, proofs	

Bibliography

1. Andrew Troelsen, Phil Japikse: Pro C# 10 with .NET 6
2. Robert C. Martin: Clean code
3. Robert C. Martin: Clean architecture
4. Roy Osherove: The art of unit testing
5. Scott Chacon: Pro Git
6. Adam Freeman: Pro ASP.NET Core 6

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 respects the IEEE and ACM Curricula Recommendations for Computer Science Studies;
- The course exists in the studying program of all major universities in Romania and abroad;
- The content of the course contains knowledge mandatory for any IT specialist working in a software company

10. Evaluation

Type of activity	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Share in the grade (%)
10.4 Course	Know the presented concepts & SE principles	Written exam	50%
10.5 Seminar/lab activities	Be able to implement acknowledged knowledge in producing software	Continuous observations	50%
10.6 Minimum performance standards			
➢ Both written exam and laboratory activity average need to be at least 6.00			

Date	Signature of course coordinator	Signature of seminar coordinator
04/03/2024	..........[illegible]..........	..........[illegible]..........

Date of approval	Signature of the head of department
....	conf. dr. Adrian Sterca