SYLLABUS

Software engineering

University year 2024-2027

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
1.7. Form of education	

2. Information regarding the discipline

2.1. Name of the discipline	**Software engineering**					Discipline code	MLE5011
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. Discipline regime	DF

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	**3**
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)					24
Additional documentation (in libraries, on electronic platforms, field documentation)					12
Preparation for seminars/labs, homework, papers, portfolios and essays					24
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	Advance programming methods, Databases
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	Projector

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	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

6.2. Learning outcomes

Knowledge	The student knows: industry standard techniques for designing, implementing, testing, and maintaining complex software systems. Acquired knowledge includes project life cycle, a small number of uml diagrams, requirements gathering, working with tasks, using git in a team, expanded C# with .NET knowledge, desktop development with WinUI, and web development with ASP.NET (MVC & API), entity framework use, regular and mocked unit tests, several design patterns, and architectural patterns
Skills	The student is able to work in small to medium sized teams (6-30 team members). Student acquired or expanded upon the following skills: Requirements engineering, UML, Working in teams, Git, GUI/Design, Software architectures, C# + .NET, Code quality, Communication skill, Unit testing, Database access, Web development, Design patterns, Deployment, Clean code
Responsibility and autonomy:	The student has the ability to work in a team to craft quality software.

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 software architectures

8. Content

8.1 Course	Teaching methods	Remarks
1. Introduction to Software engineering	Exposure: description, explanation,	

	examples, discussion of case studies	
2. Software design	Exposure: description, explanation, examples, discussion of case studies	
3. Working in teams	Exposure: description, explanation, examples, discussion of case studies	
4. C# + .NET with WinUI	Exposure: description, explanation, examples, discussion of case studies	
5. Code quality	Exposure: description, explanation, examples, discussion of case studies	
6. Unit testing	Exposure: description, explanation, examples, discussion of case studies	
7. Design patterns	Exposure: description, explanation, examples, discussion of case studies	
8. Dependency management	Exposure: description, explanation, examples, discussion of case studies	
9. ASP.NET MVC	Exposure: description, explanation, examples, discussion of case studies	
10. Working with servers	Exposure: description, explanation, examples, discussion of case studies	
11. Software projects and project management	Exposure: description, explanation, examples, discussion of case studies	
12. Software architectures	Exposure: description, explanation, examples, discussion of case studies	
13. Hiring process	Exposure: description, explanation, examples, discussion of case studies	
14. Exam	Exposure: description, explanation, examples, discussion of case studies	

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. Martin Fowler: Patterns of Enterprise Application Architecture
7. Bruce M. Van Horn II: Real-World Implementation of C# Design Patterns
8. Adam Freeman: Pro ASP.NET Core 6
9. Konstantin Semenenko: C# Interview Guide

8.2 Seminar	Teaching methods	Remarks
1. Requirements gathering	Explanation, Dialogue, debate, case studies, examples, proofs	
2. Paper prototyping	Explanation, Dialogue, debate, case studies, examples, proofs	
3. Git use in teams	Explanation, Dialogue, debate, case studies, examples, proofs	
4. Generative AI use	Explanation, Dialogue, debate, case studies, examples, proofs	
5. Deploy to server	Explanation, Dialogue, debate, case studies, examples, proofs	
6. Code generation	Explanation, Dialogue, debate, case studies, examples, proofs	
7. Project support	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. Planning initial version of project with the use of uml and project management techniques	Explanation, Dialogue, debate, case studies, examples, proofs	
3. Software development in teams	Explanation, Dialogue, debate, case studies, examples, proofs	
4. Code review, refactoring, unit and integration testing	Explanation, Dialogue, debate, case studies, examples, proofs	
5. Client server development in larger teams	Explanation, Dialogue, debate, case studies, examples, proofs	
6. Web development	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. Martin Fowler: Patterns of Enterprise Application Architecture
7. Bruce M. Van Horn II: Real-World Implementation of C# Design Patterns
8. Adam Freeman: Pro ASP.NET Core 6
9. Konstantin Semenenko: C# Interview Guide

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

Activity type	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Percentage of final grade
10.4 Course	Know the presented concepts & SE principles	Team project	0%
10.5 Seminar/laboratory	Be able to implement acknowledged knowledge in producing software	Team project	100%
10.6 Minimum standard of performance			
•			

11. Labels ODD (Sustainable Development Goals)

Not applicable.

Date:
01/05/2025

Signature of course coordinator

....

Signature of seminar coordinator

....

Date of approval:
...

Signature of the head of department

Assoc.prof.phd. Adrian STERCA