SYLLABUS

Software Design

University year 2025 - 2026

1. Information regarding the programme

1.1. Higher education institution	Babeş-Bolyai University of Cluj-Napoca
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	Master
1.6. Study programme/Qualification	Software Engineering
1.7. Form of education	Full time

2. Information regarding the discipline

2.1. Name of the discipline	Software Design					Discipline code	**MME8065**
2.2. Course coordinator					Assoc. Prof. PhD. Molnar Arthur-Jozsef		
2.3. Seminar coordinator					Assoc. Prof. PhD. Molnar Arthur-Jozsef		
2.4. Year of study	**1**	2.5. Semester	**2**	2.6. Type of evaluation	**E**	2.7. Discipline regime	**Elective**

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

3.1. Hours per week	**3**	of which: 3.2 course	**2**	3.3 seminar/laboratory/project	**1**
3.4. Total hours in the curriculum	**42**	of which: 3.5 course	**28**	3.6 seminar/laboratory/project	**14**
Time allotment for individual study (ID) and self-study activities (SA)					**hours**
Learning using manual, course support, bibliography, course notes (SA)					30
Additional documentation (in libraries, on electronic platforms, field documentation)					40
Preparation for seminars/labs, homework, papers, portfolios and essays					40
Tutorship					10
Evaluations					10
Other activities:					3
3.7. Total individual study hours	**133**				
3.8. Total hours per semester	**175**				
3.9. Number of ECTS credits	**7**				

4. Prerequisites (if necessary)

4.1. curriculum	-
4.2. competencies	Programming skills in at least one language that supports the object-oriented paradigm; knowledge and competences regarding the important phases of the software development lifecycle.

5. Conditions (if necessary)

5.1. for the course	Classroom with video-projector and Internet access.
5.2. for the seminar /lab activities	Classroom with video-projector and Internet access.

6.1. Specific competencies acquired [1]

Professional/essential competencies	• Analysis, design, and implementation of software systems • Proficient use of methodologies and tools specific to programming languages and software systems
Transversal competencies	• Teamwork capabilities; able to fulfil different roles • Professional communication skills; concise and precise description, both oral and written, of professional results, negotiation abilities

6.2. Learning outcomes

Knowledge	• The graduate has the necessary knowledge to devise, model and design complex software applications • The graduate possesses the fundamental knowledge for modelling, being able to analyse real life problems and translate them in concrete requirements and design a corresponding software model
Skills	• The graduate has the ability to follow the entire life cycle of software system development • The graduate can use specific language and terminology for software engineering and is able to communicate and interact with members of a team
Responsibility and autonomy:	• The graduate is able to carry out activities for education and training on different topics related to software development • The graduate can apply advanced information system knowledge starting from a high level of abstraction and is able to offer implementation solutions for complex software systems

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

7.1 General objective of the discipline	Gain knowledge and competences with real-world applicability regarding the design, development and maintenance of modern, complex software systems.
7.2 Specific objective of the discipline	• Know and understand the fundamental concepts of software design. • Know the most common software system types and the recommended architectural styles and design patterns employed in their development. • Have knowledge of and be able to apply the appropriate architectural and design patterns to different programming projects.

[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.

8. Content

8.1 Course	Teaching methods
Introduction The Software Development Lifecycle and the Software Process; Practical example – challenges in scaling an application from 1 to 1 million users	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration
Challenges in software development Requirements volatility, process, technology, ethical and professional practices, managing design influences. Practical examples from major companies and platforms	
The software development lifecycle Requirements, software architecture, detailed design, construction design, human-computer interface design, software design documentation and management.	
Patterns and styles in software architecture Layered, client-server, peer-to-peer, MVC, broker, blackboard, master-slave, service-oriented architecture, microservices, blockchain and smart contracts.	
Establishing system architecture	
Establishing the technology stack	
Presentation of case study systems Large-scale system architecture such as those for media streaming, social media, document and e-mail management.	
Construction and detailed design SOLID principles, component design principles, design patterns.	
Software security Discussion and presentation of hardware and software vulnerabilities in real-life systems, introduction to security and privacy risk analysis, threat modelling, the MITRE ATT&CK database, the Common Vulnerabilities and Exposures database.	
Software quality and maintenance Software quality standards and tools, antipatterns, code smells, technical debt and refactoring.	

Bibliography
1. E. Gamma, R. Helm, R.Johnson, J. Vlissides – ***Design Patterns: Elements of Reusable Object-Oriented Software***, Addison Wesley, 1995.
2. Eric Freeman, Elisabeth Robson, Bert Bates, Kathy Sierra - ***Head First Design Patterns***, O'Reilly Media, 2004.
3. William J. Brown, Raphael C. Malveau, Hays W. "Skip" McCormick, Thomas J. Mowbray - ***AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis***, Wiley, 1998.
4. Hohpe Gregor, Woolf Bobby - ***Enterprise Integration Patterns***, Addison-Wesley, 2003 (some resources at https://www.enterpriseintegrationpatterns.com/).
5. Martin Fowler - ***Refactoring. Improving the Design of Existing Code***. Addison-Wesley, 1999.
6. Carlos Otero - ***Software Engineering Design - Theory and Practice***, CRC Press, Taylor & Francis Group, 2012.
7. Alex Xu – ***System Design Interview – An Insider's Guide*** (Volumes I and II), ByteByteGo, 2020.
8. Martin Kleppmann - ***Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems***, O'Reilly Media, 2017.
9. Robert C. Martin - ***Clean Architecture: A Craftsman's Guide to Software Structure and Design***, Pearson, 2017.

8.2 Seminar / laboratory	Teaching methods
Introduction to course; presentation of evaluation method and semester projects.	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration
Initial discussion of the seminar project and the architecture documentation.	
Work on seminar project and architecture documentation. Student presentations.	
Presentation of the software design process in real-life applications. Student presentations.	
Evaluation of the first phase of the seminar project.	
Work on seminar project and architecture documentation. Student presentations.	
Evaluation of the final phase of the seminar project.	

Bibliography
1. E. Gamma, R. Helm, R.Johnson, J. Vlissides – ***Design Patterns: Elements of Reusable Object-Oriented Software***,

Addison Wesley, 1995.
2. Eric Freeman, Elisabeth Robson, Bert Bates, Kathy Sierra - ***Head First Design Patterns***, O'Reilly Media, 2004.
3. William J. Brown, Raphael C. Malveau, Hays W. "Skip" McCormick, Thomas J. Mowbray - ***AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis***, Wiley, 1998.
4. Hohpe Gregor, Woolf Bobby - ***Enterprise Integration Patterns***, Addison-Wesley, 2003 (some resources at https://www.enterpriseintegrationpatterns.com/).
5. Martin Fowler - ***Refactoring. Improving the Design of Existing Code.*** Addison-Wesley, 1999.
6. Carlos Otero - ***Software Engineering Design - Theory and Practice***, CRC Press, Taylor & Francis Group, 2012.
7. Alex Xu – ***System Design Interview – An Insider's Guide*** (Volumes I and II), ByteByteGo, 2020.
8. Martin Kleppmann - ***Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems***, O'Reilly Media, 2017.
9. Robert C. Martin - ***Clean Architecture: A Craftsman's Guide to Software Structure and Design***, Pearson, 2017.

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 study program of all major universities in Romania and abroad.

10. Evaluation

Activity type	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Percentage of final grade
10.4 Course	Presentation during the lecture or seminar	Quality of the presentation and the examples.	**20%**
	Create the architecture documentation for a complex software application	Quality and level of detail of the documentation.	**40%**
	Written exam during the regular or retake exam session.	Quality of the provided answers.	**10%**
10.5 Seminar/laboratory	Analyse the architecture and evolution of a complex, open-source application	Quality and level of detail of the documentation.	**30%**
10.6 Minimum standard of performance			

- Students must observe the standards of academic integrity.
- An average grade of 5.00 must be obtained from team presentation(s), the seminar projects and the written examination.

11. Labels ODD (Sustainable Development Goals)

Not applicable.

Date:
30.04.2025

Signature of course coordinator

Assoc. Prof. PhD. Molnar Arthur-Jozsef

Signature of seminar coordinator

Assoc. Prof. PhD. Molnar Arthur-Jozsef

Date of approval:
...

Signature of the head of department

Assoc.prof.phd. Adrian STERCA