SYLLABUS

1. Information regarding the programme

1.1 Higher education institution	**Babeș-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	**Master**
1.6 Study programme / Qualification	**Software Engineering**

2. Information regarding the discipline

2.1 Name of the discipline (en) (ro)			**Software Design / Proiectarea Sistemelor Software**				
2.2 Course coordinator			**Lect. PhD. Molnar Arthur-Jozsef**				
2.3 Seminar coordinator			**Lect. PhD. Molnar Arthur-Jozsef**				
2.4. Year of study	**1**	2.5 Semester	**2**	2.6. Type of evaluation	**E**	2.7 Type of discipline	**elective**
2.8 Code of the discipline		**MME8065**					

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	**1**
3.4 Total hours in the curriculum	**42**	Of which: 3.5 course	**28**	3.6 seminar/laboratory	**14**
Time allotment:					hours
Learning using manual, course support, bibliography, course notes					**35**
Additional documentation (in libraries, on electronic platforms, field documentation)					**35**
Preparation for seminars/labs, homework, papers, portfolios and essays					**35**
Tutorship					**14**
Evaluations					**14**
Other activities:					-

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	• Fundamentals of Programming • Object-Oriented Programming

	• Programming Paradigms
4.2. competencies	• Average Programming Skills

5. Conditions (if necessary)

5.1. for the course	• Video-projector, Internet access
5.2. for the seminar /lab activities	• Computers with Internet access and UML tooling

6. Specific competencies acquired

Professional competencies	• Understand the software design process from an engineering perspective • Understand the software design concepts and principles • Understand the software design process and its activities • Understand the specifics of the main architectural and design patterns and how to apply them to specific projects
Transversal competencies	• Professional communication skills; concise and precise description, both oral and written description of professional results • Independent and teamwork capabilities; able to fulfil different roles in the software development process • Entrepreneurial skills

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

7.1 General objective of the discipline	• Know and understand fundamental concepts of software design • Be able to apply the appropriate architectural and design patterns to different programming projects
7.2 Specific objective of the discipline	• Know and understand the main concepts and principles of software design • Have a good understanding of the following terms: software architecture definition(s), architectural styles and models, detailed design; design pattern, construction design • Learn the importance of architectural and detailed design • Know several software system types and the recommended architectural styles and design patterns associated with them.

8. Content

8.1 Course	Teaching methods	Remarks
1. Introduction to the Software Development Lifecycle and the Software Process	Interactive exposure, explanation, conversation, demonstration, student prepared presentations.	
2. Challenges in Software Development: Requirements Volatility, Process, Technology, Ethical and Professional Practices, Managing Design Influences		
3. The Software Development Lifecycle – Requirements		

4. The Software Development Lifecycle – Software Architecture		
5. The Software Development Lifecycle – Detailed Design and Construction Design		
6. The Software Development Lifecycle – Human Computer Interface Design		
7. The Software Development Lifecycle – Software Design Documentation and Management		
8. Patterns and Styles in Software Architecture – Layered, Client-Server		
9. Patterns and Styles in Software Architecture – Peer-to-Peer, MVC, Broker, Blackboard, Master-Slave		
10. Patterns and Styles in Software Architecture – Service Oriented Architecture, Microservices, Blockchain and Smart Contracts		
11. Establishing System Architecture and the Technology Stack		
12. Presentation of Real-Life Use Cases (Requirements, Architecture, Documentation)		
13. Deep-Dive into Construction Design (SOLID Principles, Component Design Principles, Design Patterns)		
14. Software Quality and Maintenance (Software Quality Standards and Tools, Antipatterns, Code Smells, Refactoring, Technical Debt)		

Bibliography

Design Patterns – Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides (1994)
AntiPatterns - Refactoring Software, Architectures and Projects in Crisis – William Brown et al (1998)
Enterprise Integration Patterns – Hohpe Gregor, Woolf Bobby (2003)
Head First Design Patterns – Eric Freeman, Elisabeth Robson (2004)
Object-Oriented Analysis and Design with Applications – Grady Booch et al. (2007)
Just Enough Software Architecture - A Risk-Driven Approach – George Fairbanks (2010)
Software Engineering Design - Theory and Practice – Carlos Otero (2012)
Software Engineering - A Practitioner's Approach – Roger Pressman (2014)
Clean Architecture – Robert C Martin (2017)
Refactoring – Martin Fowler (2018)

8.2 Seminar / laboratory	Teaching methods	Remarks
1. Administrative issues; presentation of the course and evaluation method.	Interactive exposure, explanation, conversation, demonstration, student prepared presentations.	
2. Initial discussion regarding the seminar project and the architectural documentation.		
3. Work on seminar project and architectural documentation.		
4. Presentation of the Software Design process in Real-Life Applications		
5. Evaluation of the first phase of the seminar project.		
6. Work on seminar project and architectural documentation.		

7. Evaluation of the final phase of the seminar project.		

Bibliography
Design Patterns – Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides (1994)
AntiPatterns - Refactoring Software, Architectures and Projects in Crisis – William Brown et al (1998)
Enterprise Integration Patterns – Hohpe Gregor, Woolf Bobby (2003)
Head First Design Patterns – Eric Freeman, Elisabeth Robson (2004)
Object-Oriented Analysis and Design with Applications – Grady Booch et al. (2007)
Just Enough Software Architecture - A Risk-Driven Approach – George Fairbanks (2010)
Software Engineering Design - Theory and Practice – Carlos Otero (2012)
Software Engineering - A Practitioner's Approach – Roger Pressman (2014)
Clean Architecture – Robert C Martin (2017)
Refactoring – Martin Fowler (2018)

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

- This course follows the IEEE and ACM Curriculla Recommendations for Software Engineering studies
- Courses with similar content are taught in the major universities in Romania offering similar study programs
- Course content is considered very important by the software companies for improving average software development skills

10. Evaluation

Type of activity	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Share in the grade (%)
10.4 Course	• Team presentation during the lecture • Create the architectural documentation for a complex software application	Written Exam Team Presentation	40% 20%
10.5 Seminar/lab activities	• Analyse the architecture and evolution of a complex open-source application	Seminar Project Attendance	30% 10%
10.6 Minimum performance standards			
➢ At least a grade of 5.			

Date	Signature of course coordinator	Signature of seminar coordinator
06.07.2023	Lect. PhD. Molnar Arthur-Jozsef	Lect. PhD. Molnar Arthur-Jozsef

Date of approval

.....

Signature of the head of department

Prof. PhD. Dioșan Laura