SYLLABUS

Programming Paradigms

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	**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	**Programming Paradigms**					Discipline code	**MME8028**
2.2. Course coordinator				Assoc.Prof.Eng. Florin Craciun			
2.3. Seminar coordinator				Assoc.Prof.Eng. Florin Craciun			
2.4. Year of study	1	2.5. Semester	1	2.6. Type of evaluation	E	2.7. Discipline regime	Compulsory

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/project	**2**
3.4. Total hours in the curriculum	56	of which: 3.5 course	28	3.6 seminar/laboratory/project	**28**

Time allotment for individual study (ID) and self-study activities (SA)	**hours**
Learning using manual, course support, bibliography, course notes (SA)	28
Additional documentation (in libraries, on electronic platforms, field documentation)	28
Preparation for seminars/labs, homework, papers, portfolios and essays	35
Tutorship	14
Evaluations	14
Other activities:	

3.7. Total individual study hours	**119**
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 • Functional and Logic Programming
4.2. competencies	Average software development skills

5. Conditions (if necessary)

5.1. for the course	projector
5.2. for the seminar /lab activities	projector

6.1. Specific competencies acquired [1]

Professional/essential competencies	• understanding and working with basic concepts in software engineering; • capability of analysis and synthesis; • modeling and solving real-life problems; • assimilation of mathematical concepts and formal models to understand, verify and validate software systems; • analysis, design, and implementation of software systems; • proficient use of methodologies and tools specific to programming languages and software systems;
Transversal competencies	• team work capabilities; able to fulfill different roles; • professional communication skills; concise and precise description, both oral and written, of professional results , negociation abilities;

6.2. Learning outcomes

Knowledge	The student knows: • The graduate has the necessary knowledge to devise, model and design of complex software application • The graduate knows the software processes and can integrate them in the organisational culture of a software company • The graduate possesses the fundamental knowledge for modelling, being able to analyse real life problems and to translate them in concrete requirements and to design a corresponding software model
Skills	The student is able to • The graduate proves advance programming skills which will allow to learn and comprehend modern technologies • The graduate can apply advanced information system knowledge starting from a high level of abstraction and being able to offer implementation solutions for complex software system • The graduate can use specific language and terminology for software engineering being able to communicate and interact with members of a team
Responsibility and autonomy:	The student has the ability to work independently to obtain • The graduate has the ability to combine information in different ways in order to form a positive attitude towards his/her own development • The graduate is able to carry on activities for education and training on different topics related to software development • The graduate is able to analyse concrete educational situation in terms of general ethical principles and rules • The graduate proves knowledge related to specifying the requirements of research activities in the domain of computer science in general and software engineering in particular and he/she understands the role of research in promoting progress

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

7.1 General objective of the discipline	• Know and understand fundamental concepts of programming. • Be able to apply different programming paradigms 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.

7.2 Specific objective of the discipline	• know the main features of different programming paradigms: procedural, object-oriented, concurrent, functional, logical, event-based, scripting • have a good understanding of the following concepts: value, type, variable, binding, procedural abstraction, data abstraction, object, class, component, interface, polymorphism; • learn the similarities and differences between different programming paradigms in terms of the concepts they implement

8. Content

8.1 Course	Teaching methods	Remarks
1. Basic concepts	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
2. Oz syntax, data structures		
3. Oz syntax, data structures		
4. Statements, Kernel Language, Abstract Machine		
5. Higher-Order Programming		
6. Lambda Calculus		
7. Tupled Recursion and Exceptions		
8. Types, ADT, Components		
9. Declarative Concurrency		
10. Declarative Concurrency		
11. Declarative Concurrency		
12. Stateful Programming		
13. Relational Programming		
14. Constraint Programming		

Bibliography

1. SCOTT, MICHAEL L.: Programming Language Pragmatics, 4th ed, Morgan-Kaufmann, 2016
2. SEBESTA, ROBERT W.: Concepts of Programming Languages, 10th ed, Pearson Education, 2012
3. SZYPERSKI, CLEMENS: Component Software. Beyond Object-Oriented Programming, Addison Wesley (1st ed. 1998, 2nd ed. 2002 with GRUNTZ, DOMINIK and MURER, STEFAN).
4. STROUSTRUP, BJARNE: The C++ Programming Language Special Edition, Addison-Wesley, 2000 chapter 2
5. VAN ROY, PETER; HARIDI, SEIF: Concepts, Techniques and Models of Computer Programming, MIT Press, 2004
6. WATT, David A.: Programming Language Design Concepts, Wiley, 2004

8.2 Seminar / laboratory	Teaching methods	Remarks
1. Research papers allocation for the oral presentation	Use practical tools to implement group projects. Discuss research papers.	Seminar is organized as a total of 14 hours – 2 hours every second week Project is organized as a total of 14 hours – 2 hours every
2. First programming assignment		
3. Research papers presentations		
4. Second programming assignment		
5. Research papers presentations.		
7. Third programming assignment		
8. Research papers presentations.		

Bibliography

Research papers
Mozart System

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 Curriculla Recommendations for Software Engineering studies;
- The content of the course is considered by the software companies as important for average software development skills

10. Evaluation

Activity type	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Percentage of final grade
10.4 Course	- know the basic principle of the domain; - apply the course concepts problem solving	Final written exam	40%
10.5 Seminar/laboratory	- be able to apply course concepts be able to do a critical evaluation of research papers	-Paper presentation and approx. 3 programming assignments	60%
10.6 Minimum standard of performance			

- At least grade 5 (from a scale of 1 to 10) at both final written exam and seminar work.

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

Not applicable.

Date:
...

Signature of course coordinator

Assoc.Prof.Eng. Florin Craciun

Signature of seminar coordinator

Assoc.Prof. Eng. Florin Craciun

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*.".