SYLLABUS

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	**Bachelor**
1.6 Study programme / Qualification	**Computer Science**

2. Information regarding the discipline

2.1 Name of the discipline (en) (ro)			**Pragmatic issues in programming (Aspecte pragmatice în programare)**				
2.2 Course coordinator			**Lect. PhD. Radu Lupsa**				
2.3 Seminar coordinator			**Lect. PhD. Radu Lupsa**				
2.4. Year of study	**3**	2.5 Semester	**5**	2.6. Type of evaluation	**C**	2.7 Type of discipline	**Optional**
2.8 Code of the discipline		MLE5056					

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	1L + 1P
3.4 Total hours in the curriculum	56	Of which: 3.5 course	28	3.6 seminar/laboratory	28

Time allotment:	hours
Learning using manual, course support, bibliography, course notes	10
Additional documentation (in libraries, on electronic platforms, field documentation)	10
Preparation for seminars/labs, homework, papers, portfolios and essays	20
Tutorship	2
Evaluations	2
Other activities:	

3.7 Total individual study hours	44
3.8 Total hours per semester	100
3.9 Number of ECTS credits	4

4. Prerequisites (if necessary)

4.1. curriculum	• Advanced programming methods
4.2. competencies	• Average skills in programming.

5. Conditions (if necessary)

5.1. for the course	•
5.2. for the seminar /lab activities	• Laboratory with computers; high level programming language environment (C++, Java, .NET, python)

6. Specific competencies acquired

Professional competencies	C2.1 Identificarea de metodologii adecvate de dezvoltare a sistemelor software C2.3 Utilizarea metodologiilor, mecanismelor de specificare ?i a mediilor de dezvoltare pentru realizarea aplica?iilor informatice
Transversal competencies	CT1 Aplicarea regulilor de munca organizata si eficienta, a unor atitudini responsabile fata de domeniul didactic-stiintific, pentru valorificarea creativa a propriului potential, cu respectarea principiilor si a normelor de etica profesionala CT3 Utilizarea unor metode si tehnici eficiente de învatare, informare, cercetare si dezvoltare a capacitatilor de valorificare a cunostintelor, de adaptare la cerintele unei societati dinamice ?i de comunicare în limba româna ?i într-o limba de circula?ie interna?ionala

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

7.1 General objective of the discipline	• General improvement of programming efficiency. • Approach programming from a practical point of view.
7.2 Specific objective of the discipline	• Improve programming efficiency by using a disciplined approach; • Be aware of the time-consuming tasks while programming and the tools and methods to avoid them.

8. Content

8.1 Course	Teaching methods	Remarks
1. Development speed, long-term versus short-term speed. Complexity as the main asymptotic slow-down factor. The role of a disciplined, systematic approach.	Interactive exposure Explanation Conversation Didactical demonstration	
2. Programming discipline: Tracking changes and (automated) testing: goals, issues, best practices.	Interactive exposure Explanation Conversation Didactical demonstration	
3. Programming discipline: *One Responsibility Rule* principle, *Don't Repeat Yourself*	Interactive exposure Explanation	

principle, Coupling and cohesion. Refactoring.	Conversation Didactical demonstration	
4. Programming discipline: code documentation. Pre/post conditions, border cases, well-chosen identifiers, tools.	Interactive exposure Explanation Conversation Didactical demonstration	
5. Programming discipline: Undefined behaviour, implementation defined behaviour, premature optimization, good optimization.	Interactive exposure Explanation Conversation Didactical demonstration	
6. Programming discipline: defensive programming. assert() on pre/post conditions and invariants. Input data validation. Fail fast principle.	Interactive exposure Explanation Conversation Didactical demonstration	
7. Programming discipline: Input data validation, efficient diagnosing of errors, secure code.	Interactive exposure Explanation Conversation Didactical demonstration	
8. Testing and debugging techniques: IDE debugger, assert(), core dumps, regression tests, logging and log filtering.	Interactive exposure Explanation Conversation Didactical demonstration	
9. Patterns and techniques: Classes: value semantic vs. object semantic. Immutable classes.	Interactive exposure Explanation Conversation Didactical demonstration	
10. Patterns and techniques: Constructors, destructors, resources and invariants. RAII.	Interactive exposure Explanation Conversation Didactical demonstration	
11. Patterns and techniques: exceptions. Exception safety levels.	Interactive exposure Explanation Conversation Didactical demonstration	
12. Patterns and techniques: multi-threading patterns.	Interactive exposure Explanation Conversation Didactical demonstration	
13. Source control tools and best practices	Interactive exposure Explanation Conversation	

	Didactical demonstration	
14. Continous integration tools and best practices	Interactive exposure Explanation Conversation Didactical demonstration	

Bibliography

1. Michael Howard and David LeBlanc: *Writing Secure Code*, MicrosoftPress, 2003.

2. Herb Sutter, Andrei Alexandrescu: *C++ Coding Standards: 101 Rules, Guidelines, and Best Practices.* Addison-Wesley, 2010.

3. Martin Fowler and others: *Refactoring: Improving the Design of Existing Code.* Addison-Wesley, 1999.

4. Robert C. Martin: *Clean Code: A Handbook of Agile Software Craftsmanship.* Prentice Hall.

5. Andrew Hunt , David Thomas: *The Pragmatic Programmer: From Journeyman to Master.* Addison-Wesley, 2000.

6. Marshall P. Cline, Greg Lomow, Mike Girou: *C++ FAQs (2nd Edition)*. Addison-Wesley, 1999.

8.2 Seminar / laboratory	Teaching methods	Remarks
1. Introduction, administrative issues. Code examples.	Dialogue, debate, case study, guided discovery	
2. Programming discipline: One Responsibility Rule principle, Don’t Repeat Yourself principle, Coupling and cohesion. Refactoring. Code documentation. Pre/post conditions, border cases, well-chosen identifiers, tools.	Dialogue, debate, case study, guided discovery	
3. Programming discipline: Undefined behaviour, implementation defined behaviour, premature optimization, good optimization. Defensive programming. assert() on pre/post conditions and invariants. Input data validation. Fail fast principle.	Dialogue, debate, case study, guided discovery	
4. Programming discipline: Input data validation, efficient diagnosing of errors, secure code. Testing and debugging techniques: IDE debugger, assert(), core dumps, regression tests, logging and log filtering.	Dialogue, debate, case study, guided discovery	
5. Patterns and techniques: Classes: value semantic vs. object semantic. Immutable classes. Constructors, destructors, resources and invariants. RAII.	Dialogue, debate, case study, guided discovery	
6. Patterns and techniques: exceptions. Exception safety levels. Multi-threading patterns.	Dialogue, debate, case study, guided discovery	
7. Programming discipline: Tracking changes and (automated) testing.	Dialogue, debate, case study, guided discovery	

Bibliography

7. Michael Howard and David LeBlanc: *Writing Secure Code*, MicrosoftPress, 2003.
8. Herb Sutter, Andrei Alexandrescu: *C++ Coding Standards: 101 Rules, Guidelines, and Best Practices.* Addison-Wesley, 2010.
9. Martin Fowler and others: *Refactoring: Improving the Design of Existing Code.* Addison-Wesley, 1999.
10. Robert C. Martin: *Clean Code: A Handbook of Agile Software Craftsmanship.* Prentice Hall.
11. Andrew Hunt , David Thomas: *The Pragmatic Programmer: From Journeyman to Master.* Addison-Wesley, 2000.
12. Marshall P. Cline, Greg Lomow, Mike Girou: *C++ FAQs (2nd Edition)*. Addison-Wesley, 1999.

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 content of the course comes from practical field experience.

10. Evaluation

Type of activity	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Share in the grade (%)
10.4 Course			
10.5 Seminar/lab activities	- know the basic principles discussed at the course and know to apply them; - recognize the weak spots in a program; - find good ways to avoid the weak spots	Verifying the practical works.	50%
	- be able to show the understanding of the principles in a mini-project	Verifying the project	
10.6 Minimum performance standards			
➢ At least grade 5 (from a scale of 1 to 10) for the average.			

Date	Signature of course coordinator	Signature of seminar coordinator
....	....	....

Date of approval	Signature of the head of department

....……............................