SYLLABUS

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**

2. Information regarding the discipline

2.1 Name of the discipline	**Automated Program Analysis**						
2.2 Course coordinator	**Assoc. Prof. Florin Craciun**						
2.3 Seminar coordinator	**Assoc. Prof. Florin Craciun**						
2.4. Year of study	**3**	2.5 Semester	**6**	2.6. Type of evaluation	**E**	2.7 Type of discipline	**OPTIONAL**

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	48	Of which: 3.5 course	28	3.6 seminar/laboratory	14
Time allotment:					hours
Learning using manual, course support, bibliography, course notes					8
Additional documentation (in libraries, on electronic platforms, field documentation)					7
Preparation for seminars/labs, homework, papers, portfolios and essays					8
Tutorship					2
Evaluations					8
Other activities:					-
3.7 Total individual study hours	33				
3.8 Total hours per semester	75				
3.9 Number of ECTS credits	5				

4. Prerequisites (if necessary)

4.1. curriculum	Fundamentals of Programming, Algorithms and Data Structures, Object-Oriented Programming, Advanced Programming Methods
4.2. competencies	Basic knowledge in Python, Java, C#, C++

5. Conditions (if necessary)

5.1. for the course	Projector for lecture presentations
5.2. for the seminar /lab activities	Computers for practical assignments

6. Specific competencies acquired

Professional competencies	• Good programming skills in high-level languages • Better understanding of the program execution • Better knowledge about automated program verification • Better knowledge about writing correct code • Better knowledge about code optimization
Transversal competencies	• Ability to design and build dependable software systems

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

7.1 General objective of the discipline	• Understanding of the main concepts and techniques to automatically analise, optimize and verify the programs
7.2 Specific objective of the discipline	• To understand the execution model of a program • To understand the automated program analyse • To understand the automated techniques to optimized the program • To understand the automated program verification • To become familiar with the tools which automatically analise, optimize and verify the programs

8. Content

8.1 Course	Teaching methods	Remarks
• Operational Semantics for a simple imperative language	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
• Program Static Analysis	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
• Automated Optimization Techniques for programs	• Interactive exposure • Explanation	

	• Conversation • Didactical demonstration	
• DataFlow Analysis -Part 1	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
• DataFlow Analysis -Part 2	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
• Control Flow Analysis	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
• Symbolic Execution of a program	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
• Hoare Logic -part1	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
• Hoare Logic -part2	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
• Pointer Analysis	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
• Separation Logic- part1	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
• Separation Logic -part2	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
• Abstract Interpretation – Part 1	• Interactive exposure • Conversation	
• Abstract Interpretation-part2	• Interactive exposure • Conversation	

Bibliography 1. F. Nielson, H.R. Nielson, C. Hankin, Principles of Program Analysis		
8.2 Seminar / laboratory	Teaching methods	Remarks
1. Tools to visualize operational semantics	Conversation, debate, case studies, examples	The laboratory is structured as 2 hours classes every second week
2. Tools for Dataflow Analysis	•	
3. Tools for Control Flow Analysis	•	
4. Tools for Symbolic Execution		
5. Tools for Automated Verification (Hoare Logic)	•	
6. Tools for Automated Verification (Separation Logic)	•	
7. Tools for Automated Verification(Separation Logic)	•	
	•	
	•	
Bibliography The latest academic tools open source. The students will be able to change/adapt the tools.		

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 Computer Science studies

- The content of the course is considered by the software companies as important for average software development skills

10. Evaluation

Course	• - know the basic principle of • the domain; • - apply the course concepts problem solving	Written Final Exam	50.00%
	•		
	•		
Seminar/lab activities	• - be able to use course • concepts in solving the real problems	Laboratory Assignments	50.00%
	•		
• At least grade 5 (from a scale of 1 to 10) at written final exam and at each laboratory assignment.			

Date	Signature of course coordinator	Signature of seminar coordinator
	Assoc. Prof. Florin Craciun	Assoc. Prof. Florin Craciun

Date of approval	Signature of the head of department