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	**Master**
1.6 Study programme / Qualification	**High Performance Computing and Big Data Analytics Profile**

2. Information regarding the discipline

2.1 Name of the discipline (en) (ro)	**Resource Aware Computation** **Calcul Sensibil la Consumul de Resurse**						
2.2 Course coordinator	**Assoc. Prof. Eng. Florin Craciun**						
2.3 Seminar coordinator	**Assoc. Prof. Eng. Florin Craciun**						
2.4. Year of study	**2**	2.5 Semester	**3**	2.6. Type of evaluation	**E**	2.7 Type of discipline	**DS**

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

3.1 Hours per week	5	Of which: 3.2 course	2	3.3 seminar/laboratory	3
3.4 Total hours in the curriculum	70	Of which: 3.5 course	28	3.6 seminar/laboratory	42

Time allotment:	hours
Learning using manual, course support, bibliography, course notes	30
Additional documentation (in libraries, on electronic platforms, field documentation)	10
Preparation for seminars/labs, homework, papers, portfolios and essays	98
Tutorship	10
Evaluations	10
Other activities:	-

3.7 Total individual study hours	158
3.8 Total hours per semester	200
3.9 Number of ECTS credits	8

4. Prerequisites (if necessary)

4.1. curriculum	• None
4.2. competencies	• Basic software development skills • Procedural and Object-oriented paradigms

5. Conditions (if necessary)

5.1. for the course	

6. Specific competencies acquired

Professional competencies	• Understanding and working with basic concepts in software engineering; • Knowledge, understanding and use of basic concepts of theoretical Computer Science • Capability of analysis and synthesis; • Proficient use of methodologies and tools specific tool software systems • Good programming skills in high-level languages
Transversal competencies	• Improved programming abilities: debugging and correcting compilers errors • Ability to apply compiler techniques to different real life problems

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

7.1 General objective of the discipline	• To understand fundamental concepts of software quality. • To be able to apply basic methods for software analysis and software quality assurance.
7.2 Specific objective of the discipline	• To learn the methods of program verification and validation. • To become used with building correct programs from specifications • To acquire a modern programming style • To understand how the resources(memory, CPU, batery) are used by the programs

8. Content

8.1 Course	Teaching methods	Remarks
1. Semantics of sequential programs. Procedural paradigm. Object-oriented paradigm. Functional paradigm. Operational semantics. Small-Steps Semantics. Big-Steps Semantics	Exposure,description, explanation, debate and dialogue, discussion of case studies	
2. Semantics of concurrent programs. Concurrency models. Processes & threads Atomic actions, interleaving model. Safety and liveness	explanation, debate and dialogue, discussion of case studies	
3. Semantics of multicore programs. Cell processors. Parallel architectures. Parallel programming concepts	Exposure,description, explanation	
4. Program Analysis. Principles.	Exposure,description, explanation	
5. Dataflow Analysis	Exposure,description, explanation, discussion of case studies	
6. Type-based Analysis	Exposure,description, explanation,	

	discussion of case studies	
7. Shape Analysis. Abstract Interpretation	Exposure,description, explanation,	
8. Automatic verification. Hoare logic. Separation Logic	Exposure,description, explanation	
9. Analysis and verification of memory usage. Memory models. Shape analysis	Exposure,description, explanation, discussion of case studies	
10. Analysis and verification of memory usage. Separation logic methods	Exposure,description, explanation, discussion of case studies	
11. Analysis and verification of cpu usage	Exposure,description, explanation, discussion of case studies	
12. Analysis and verification of cpu usage. Cost analysis.	Exposure,description, explanation, discussion of case studies	
13. Analysis and verification of battery usage. Energy-aware programming techniques. Approximate computations.	Exposure,description, explanation, discussion of case studies	
14. Analysis and verification of battery usage. Techniques to control the battery usage	Exposure,description, explanation, discussion of case studies	

Bibliography

1. Flemming Nielson, Hanne Riis Nielson, Chris Hankin: Principles of ProgramAnalysis, Springer, 1999.
2. Advanced Compiler Design and Implementation, by Muchnick. Morgan Kaufmann 1997
3. Benjamin C. Pierce. Types and Programming Languages
4. Neil D. Jones, Flemming Nielson. Abstract Interpretation: a Semantic-Based Tool for Program Analysis, in: Handbook of logic in computer science (vol. 4). Oxford University Press, 1995

8.2 Seminar / laboratory	Teaching methods	Remarks
1. Project 1-part1: Implement a dataflow analysis for a sequential object-oriented language	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 every week.
2. Project 1-part2: Implement a dataflow analysis for a sequential object-oriented language. Seminar 1: Discuss papers about dataflow	Use practical tools to implement group projects. Discuss research papers.	

analyses		
3. Project 1-part 3: Implement a dataflow analysis for a sequential object-oriented language.	Use practical tools to implement group projects. Discuss research papers.	
4. Project 1-part 4: Implement a dataflow analysis for a sequential object-oriented language. Seminar 2: Discuss papers about dataflow analyses	Use practical tools to implement group projects. Discuss research papers.	
5. Project 1-part 5: Implement a dataflow analysis for a sequential object-oriented language	Use practical tools to implement group projects. Discuss research papers.	
6. Project 2-part 1: Implement a type system for a sequential object-oriented language. Seminar 3: Discuss papers about type based analyses	Use practical tools to implement group projects. Discuss research papers.	
7. Project 2-part 2: Implement a type system for a sequential object-oriented language	Use practical tools to implement group projects. Discuss research papers.	
8. Project 2-part 3: Implement a type system for a sequential object-oriented language. Seminar 4: Discuss papers about type based analyses	Use practical tools to implement group projects. Discuss research papers.	
9. Project 2-part 4: Implement a type system for a sequential object-oriented language	Use practical tools to implement group projects. Discuss research papers.	
10. Project 3-part 1: Use Hip/sleek to verify the memory consumption Seminar 5: Discuss papers about separation logic	Use practical tools to implement group projects. Discuss research papers.	
11. Project 3-part 2: Use Hip/sleek to verify the memory consumption	Use practical tools to implement group projects. Discuss research papers.	
12. Project 3-part 3: Use Hip/sleek to verify the memory consumption Seminar 6: Discuss papers about separation logic	Use practical tools to implement group projects. Discuss research papers.	
13. Project 3-part 4: Use Hip/sleek to verify the memory consumption	Use practical tools to implement group projects. Discuss research papers.	
14. Project 3-part 5: Use Hip/sleek to verify the memory consumption Seminar 7: Discuss papers about separation logic	Use practical tools to implement group projects. Discuss research papers.	

Bibliography - research papers – documentation of the practical tools used by the projects

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

Type of activity	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Share in the grade (%)
10.4 Course	- know the basic principle of the domain; - apply the course concepts - problem solving	Written exam	40.00%
10.5 Seminar/lab activities -	- be able to implement course concepts - be able to do a critical evaluation of research papers - to be able to write a critical essay	-Practical projects	60.00%
10.6 Minimum performance standards			
➢ At least grade 5 (from a scale of 1 to 10) at both written exam and laboratory work.			

Date Signature of course coordinator Signature of seminar coordinator

..... Assoc. Prof. Eng. Florin CRACIUN Assoc. Prof. Eng. Florin CRACIUN

Date of approval Signature of the head of department

....….....