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	**Informatica romana**

2. Information regarding the discipline

2.1 Name of the discipline (en) (ro)			**Static Program Analysis**				
2.2 Course coordinator			**Assoc.Prof.PhD. Simona Motogna**				
2.3 Seminar coordinator			**Assoc.Prof.PhD. Simona Motogna**				
2.4. Year of study	**3**	2.5 Semester	**6**	2.6. Type of evaluation	**C**	2.7 Type of discipline	**Optional**
2.8 Code of the discipline		MLE5126					

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	1lab
3.4 Total hours in the curriculum	26	Of which: 3.5 course	24	3.6 seminar/laboratory	12
Time allotment:					hours
Learning using manual, course support, bibliography, course notes					20
Additional documentation (in libraries, on electronic platforms, field documentation)					30
Preparation for seminars/labs, homework, papers, portfolios and essays					40
Tutorship					20
Evaluations					29
Other activities:					-

3.7 Total individual study hours	139
3.8 Total hours per semester	175
3.9 Number of ECTS credits	7

4. Prerequisites (if necessary)

4.1. curriculum	• Formal Languages and Compiler Design course
4.2. competencies	• Basic knowledge of front-end and back end of a compiler • Medium programming skills

5. Conditions (if necessary)

5.1. for the course	• Room with projector
5.2. for the seminar /lab activities	• Laboratory: computers and use of a programming language environment

6. Specific competencies acquired

Professional comp	C 4.1 Definition of concepts and basic principles of computer science, and of mathematical theories C 4.2 Interpretation of mathematical and computer science models (formal) C 4.4 Use of simulation to study the behaviour of models and to evaluate their performance

etencies	
Transversal competencies	**CT1** Apply rules to: organized and efficient work, responsibilities of didactical and scientifical activ capitalization of own potential, while respecting principles and rules for professional ethics **CT3** Use efficient methods and techniques for learning, knowledge gaining, and research and develop capabilities for capitalization of knowledge, accommodation to society requirements and communication in English

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

7.1 General objective of the discipline	• Be able to understand compiler design • Be able to understand static analysis concepts • Improved programming skills
7.2 Specific objective of the discipline	• Be able to apply static analysis techniques • Be able to implement static analysis techniques

8. Content

8.1 Course	Teaching methods	Remarks
1. Static Analysis Tools: an introduction: principles, goals	Exposure: description, explanation, examples, discussion of case studies	
2. Dataflow analysis	Exposure: description, explanation, examples, discussion of case studies	
3. Abstract interpretations	Exposure: description, explanation, examples, debate, dialogue	
4. Interprocedural analysis	Exposure: description, explanation, examples, proofs	
5. Symbolic execution	Exposure: description, explanation, examples, discussion of case studies	
6. Buffer Overflow Analysis	Exposure: description, explanation, examples, discussion of case studies	
7. Analysis of heap data structures	Exposure: description, explanation, examples, discussion of case studies	

8. Detecting security vulnerabilities	Exposure: description, explanation, examples, discussion of case studies	
9. Invited lecture: how are static analysis tools used in real life projects	Exposure: description, explanation, examples, discussion of case studies	
10. – 12. Project presentations	Exposure: discussion of case studies	

Bibliography

1. A.V. AHO, D.J. ULLMAN - Principles of computer design, Addison-Wesley, 1978.
2. A.V. AHO, D.J. ULLMAN - The theory of parsing, translation and compiling, Prentice-Hall, Engl. Cliffs., N.J., 1972, 1973.
3. D. GRIES - Compiler construction for digital computers,, John Wiley, New York, 1971.
4. GRUNE, DICK - BAL, H. - JACOBS, C. - LANGENDOEN, K.: Modern Compiler Design, John Wiley, 2000
5.Flemming Nielson, Hanne R. Nielson, Chris Hankin: Principles of Program Analysis. 2nd edition, Springer, 2005
6. Steven S. Muchnick, Neil D. Jones: Program Flow Analysis: Theory and Applications. Prentice Hall, 1981
7. Anders Møller and Michael I. Schwartzbach - Static Program Analysis, Lecture notes, https://cs.au.dk/~amoeller/spa/

8.2 Seminar / laboratory	Teaching methods	Remarks
1. Case study: static analysis tool (part 1): features, techniques, applicability	Laboratory assignment, case study, conversation	
2. Case study: static analysis tool (part 1): apply for a program	Laboratory assignment, case study, conversation	
3. Choice of project topic. Problem specification	Laboratory assignment, case study, conversation	
4. Project analysis and design. – documentation	Laboratory assignment, case study, conversation	
5. Project implementation and testing	Laboratory assignment, case study, conversation	
6. Project presentations	Laboratory assignment, case study, conversation	

Bibliography

1. P. Emanuelsson, U. Nilsson: A Comparative Study of Industrial Static Analysis Tools, Technical Report 2008:3, Linköping University, Sweden, 2008
2. SonarQube documentation
3. PyLint documentation
- Other static analysis 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 the software companies as important for advanced programming skills
- The course provides a good theoretical background for further research in Software Engineering and Programming Fundamentals

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	Continuous evaluation at course	10%
	- understand advanced topics in the field	Paper presentations	30%
10.5 Lab activities	- be able to implement course concepts and algorithms	Lab assignments	40%
	- apply techniques for different classes of programming languages	Use of Static Analysis Tools	20%
10.6 Minimum performance standards			
➢ At least grade 5 (from a scale of 1 to 10) at both written exam and laboratory work. ➢ To be able to use and interpret results of Static Analysis tools ➢ To be able to explain the concepts involved in static analysis methods			

Date	Signature of course coordinator	Signature of seminar coordinator
14.04.2021	Assoc.prof.PhD. Simona Motogna	Assoc.prof.PhD. Simona Motogna

Date of approval
.....

Signature of the head of department
Prof.dr. Laura Dioșan