SYLLABUS

1. Data about the program of study

1.1 Institution	The Technical University of Cluj-Napoca
1.2 Faculty	Faculty of Automation and Computer Science
1.3 Department	Computer Science
1.4 Field of study	Computer Science and Information Technology
1.5 Cycle of study	Masters
1.6 Program of study / Qualification	Artificial Intelligence and Vision / Masters
1.7 Form of education	Full time

2. Data about the subject

2.1 Subject name	***Languages and Type Systems***			Subject code	**7.2**
2.2 Course responsible / lecturer	Prof.dr.eng. Todoran Eneia-Nicolae - Eneia.Todoran@cs.utcluj.ro				
2.3 Teachers in charge of seminars / Laboratory / project	Prof.dr.eng. Todoran Eneia-Nicolae - Eneia.Todoran@cs.utcluj.ro				
2.4 Year of study	I	2.5 Semester	2	2.6 Type of assessment (E - exam, C - colloquium, V – verification)	E
2.7 Subject category	Formative category: DF – fundamental, DS – speciality, DC – complementary				DS
	Optionality: DOB – mandatory, DOP – optional (alternative), DFA – optional (free choice)				DOP

3. Estimated total time

3.1 Number of hours per week	3	of which:	Course	2	Seminars	1	Laboratory	-	Project	-
3.2 Number of hours per semester	42	of which:	Course	28	Seminars	14	Laboratory	-	Project	-
3.3 Individual study:										
(a) Manual, lecture material and notes, bibliography										15
(b) Supplementary study in the library, online and in the field										15
(c) Preparation for seminars/laboratory works, homework, reports, portfolios, essays										15
(d) Tutoring										10
(e) Exams and tests										3
(f) Other activities:										-
3.4 Total hours of individual study (suma (3.3(a)…3.3(f)))										58
3.5 Total hours per semester (3.2+3.4)										100
3.6 Number of credit points										4

4. Pre-requisites (where appropriate)

4.1 Curriculum	Programming Languages, Software Engineering – undergraduate courses
4.2 Competence	Operating with computer science and mathematical foundations

5. Requirements (where appropriate)

5.1. For the course	Blackboard, Projector, PC with internet access - interactive course
5.2. For the applications	Projector, PC with internet access, for the maximum grade the student should attend at least 70% of the seminar hours

6. Specific competence

6.1 Professional competences	• analyse software specifications • define technical requirements • analyse big data • apply ICT systems theory • design process • develop creative ideas • align software with system architectures • create software • manage system testing • perform scientific research • provide technical documentation • interpret technical requirements • conduct scholarly research • apply statistical analysis techniques • supervise software development • use software libraries • use software design patterns
6.2 Cross competences	• develop an analytical approach • taking a proactive approach • developing strategies to solve problems • being open minded • coordinate engineering teams

7. Expected Learning Outcomes

Knowledge	The student has knowledge of: • algorithms • business process modeling • computer programming • systems development life cycle • model-based systems engineering • unified modeling language (UML) • object-oriented modeling • agile project management methodology • systems design methods • process algorithm formulation • software frameworks • ICT system programming • software for integrated development environments (IDE) • apply communication skills in the technical field
Skills	The student is able to: • analyse software specifications • define technical requirements • analyse big data • apply ICT systems theory • develop creative ideas • align software with system architectures • create software applications • supervise software development • develop software prototypes • utilise computer-aided software engineering tools • develop cloud computing services • find solutions to problems

Responsibilities and autonomy	The student has the ability to work independently in order to: • develop an analytical approach • take a proactive approach • develop strategies to solve problems • be open minded • coordinate engineering teams

8. Discipline objective (as results from the *key competences gained*)

8.1 General objective	The main objective of this discipline is to provide specific knowledge and to prepare students for the use of formal and semantic models in the design and verification of computing systems. The knowledge is presented in the context of programming and specification languages and process calculi, with an emphasis on static type checking, dynamic semantics, performance modelling and formal verification of the properties of computing systems.
8.2 Specific objectives	To achieve these general objectives, students will: • Learn to specify and formally design computer languages and systems via their operational semantics and type systems • Learn to formally verify properties of languages and systems • State and prove properties of programs based on their semantics • Learn techniques for designing and verifying the properties of languages and systems (induction, fixed point semantics, bisimulation) • Learn to apply advanced design principles and paradigms • Study how semantic techniques allow solving complex problems in formal design, performance evaluation, formal verification • Learn to apply formal methods in the specification, development and verification of software systems

9. Contents

9.1 Lectures	Hours	Teaching methods	Notes
Introduction and overview of the course	2	Interactive course, exposition, examples, questions, discussions.	
Semantic styles, operational semantics	2		
Untyped lambda-calculus (ULC): syntax and evaluation relation	2		
ULC: Nameless representation of terms (De Bruijn terms)	2		
Simply typed lambda-calculus (STLC): syntax and typing relation	2		
STLC: properties of typing (progress and preservation theorems)	2		
Simple extensions: tuples, variants, general recursion, lists	2		
Semantic interpreter for STLC - Haskell implementation	2		
Introduction to process algebras and bisimulation	2		
Bisimulation and algebraic semantics: concurrency and nondeterminism	2		
Introduction to CCS (Calculus of Communicating Systems)	2		
Bisimulation and algebraic semantics in CCS (1)	2		
Bisimulation and algebraic semantics in CCS (2)	2		
Compositionality properties	2		

Bibliography:

- B. Pierce, Types and Programming Languages, MIT Press, 2002.
- B. Pierce, Advanced Topics in Types and Programming Languages, MIT Press, 2005.
- B. Pierce et al, Software Foundations, https://softwarefoundations.cis.upenn.edu, 2025.
- R. Milner. Communicating and mobile systems: the pi-calculus, Cambridge Univ. Press, 1999.
- D. Sangiorgi, Introduction to Bisimulation and Coinduction, Cambridge University Press, 2011.
- E.M. Clarke et al, Handbook of Model Checking, Springer, 2018.
- R. Harper, Practical Foundations for Programming Languages, Cambridge University Press, 2016.

- E.N. Todoran, Types and Programming Languages (lecture notes), Technical University of Cluj-Napoca, https://ftp.utcluj.ro/pub/users/gc/LST/tpl-2025.pdf, 2025.
- E.N. Todoran. Introducere in Semantica Limbajelor de Programare, Note de curs si seminar, Universitatea Tehnica din Cluj-Napoca, http://users.utcluj.ro/~eneia/aplc-2016.pdf, 2016.

9.2 Applications - Seminars/Laboratory/Project	Hours	Teaching methods	Notes
Structural operational semantics	2	Exposition, examples, questions, discussions.	
Semantic design with transition systems	2		
Semantic design techniques: direct and continuation semantics	2		
Nameless representation of lambda-terms: substitution and beta-reduction	2		
Design and Haskell implementation of a semantic interpreter for STLC	2		
Concurrent systems verification with process algebras	2		
Algebraic laws for CCS in bisimulation semantics	2		

Bibliography

- B. Pierce, Types and Programming Languages, MIT Press, 2002.
- B. Pierce et al, Software Foundations, https://softwarefoundations.cis.upenn.edu, 2025.
- R. Milner. Communicating and mobile systems: the pi-calculus, Cambridge Univ. Press, 1999.
- E.M. Clarke et al, Handbook of Model Checking, Springer, 2018.
- D. Sangiorgi, Introduction to Bisimulation and Coinduction, Cambridge University Press, 2011.
- E.N. Todoran, Types and Programming Languages (lecture notes), Technical University of Cluj-Napoca, https://ftp.utcluj.ro/pub/users/gc/LST/tpl-2025.pdf, 2025.
- PRISM probabilistic model checker www.prismmodelchecker.org

**Se vor preciza, după caz: tematica seminariilor, lucrările de laborator, tematica și etapele proiectului.*

10. Bridging course contents with the expectations of the representatives of the community, professional associations and employers in the field

This course introduces basic knowledge in the fields of Semantics, Type Systems and Formal Methods. The presentation is made in the context of programming and specifications languages and process calculi. The languages and systems under consideration are described mathematically using formal syntax and are endowed with formal (static and dynamic) semantics. From an engineering perspective, this discipline offers important knowledge for the development of computing systems that impose strict quality standards: reliability, operational safety, measurable performance, etc. Each student must elaborate a scientific paper (an essay or a technical report). For the elaboration of the paper students can choose from a wide variety of topics: advanced topics in types and programming languages, model checking, dependent types, session types, runtime verification, nature inspired models of computation (DNA computing, membrane computing), process calculi, etc. The content of the discipline is synchronized with the latest advances in the field, based on monographs, studies and courses taught at prestigious universities in Europe and the USA.

11. Evaluation

Activity type	Assessment criteria	Assessment methods	Weight in the final grade
Course	The ability to solve problems specific to the field, attendance and activity during the course hours	Written exam - summative	75%
Seminar	The ability to solve problems specific to the field, attendance and activity in seminar classes	Elaboration of a scientific paper, seminar assignments - continuous	25%
Laboratory	-	-	-
Project	-	-	-
Minimum standard of performance: Modeling and solving semantic design problems, by using the formal apparatus specific to the field (type systems, operational semantics, bisimulation).			

Date of filling in: 29.04.2026	Responsible	Title First name Last name	Signature
	Course	Prof.dr.eng. Eneia-Nicolae TODORAN	
	Applications	Prof.dr.eng. Eneia-Nicolae TODORAN	

Date of approval in the department	Head of department, Prof.dr.eng. Rodica Potolea
Date of approval in the Faculty Council	Dean, Prof.dr.eng. Vlad Mureșan