COURSE UNIT DESCRIPTION

Course unit title	Course unit code
Type-driven development	

Lecturer(s)	Department where the course unit is delivered
Coordinator: assoc. prof. Karolis Petrauskas **Other lecturers:** p'ship assoc. prof. Viačeslav Pozdniakov	Department of Software Engineering, Institute of Compute Science, Vilnius University

Cycle	Level of course unit	Type of the course unit
First	-	Elective

Mode of delivery	Semester or period when the course unit is delivered	Language of instruction
Face-to-face	Spring semester	Lithuanian, English

Prerequisites and corequisites	
Prerequisites: "Functional programming".	**Corequisites (if any): -**

Number of ECTS credits allocated	Student's workload	Contact hours	Self-study hours
5	130	64	66

Purpose of the course unit: programme competences to be developed

Introduce students to elaborated programming language type systems, explore their application to software engineering problems, analyze relevant theoretical background, and develop skills in type-driven development and verification of software.

Generic competences:

- Continuous learning *(GC2)*.
  - Will be able to independently acquire new knowledge, methods, and tools and apply them in practice *(GC2.3)*.

Specific competences:

- Knowledge and abilities of conceptual foundations *(SC4)*:
  - Students will understand the basic concepts of software engineering, including several frontier areas, potential application domains, and the discipline's scope *(SC4.1)*.
  - Will be able to apply theoretical knowledge in mathematics, software engineering, computer science, and algorithmic principles to build software systems *(SC4.2)*.
  - Will be able to think abstractly, use formal methods, prove correctness, formalize, and model real-world problems *(SC4.3)*.
- Technological, methodological knowledge and abilities, professional competence *(SC6)*:
  - Will be able to combine the theory and practice of software engineering, considering problem-solving techniques and assessing technological, economic, social, and legal contexts *(SC6.1)*.
  - Will be able to select and use appropriate modern methods, models, problem-solving patterns, skills, and tools necessary for developing and maintaining application systems, including new application areas *(SC6.2)*.
  - Will be able to use existing computer hardware and software, identify, understand, and apply frontier technologies *(SC6.3)*.

Learning outcomes of the course unit: students will be able to	Teaching and learning methods	Assessment methods

Understand dependent types and apply them to software engineering problems.	Lectures, problem-oriented teaching, case studies, information retrieval, literary reading, individual work, tutorials, lab assignments.	Lab assignments and presentation of their results, written exam (open, semi-open and close-ended questions and tasks).
Formulate desired software properties as dependent types and develop implementations conforming to those properties.		
Analyze scientific papers in the domain of type systems and understand their contribution in the context of core knowledge.		
Apply type-driven development in the Idris2 programming language and understand conceptual relations to other languages with similar type systems.		

Course content: breakdown of the topics	**Contact hours**							**Self-study work: time and assignments**	
	Lectures	Tutorials	Seminars	Practice	Lab assignments	Practical training	**Contact hours**	**Self-study hours**	**Assignments**
1. Introduction and fundamentals.	2					2	**4**	**4**	Tasks at practical classes. Laboratory assignments. Discussion on related papers. Self-study of literature.
2. A vector as a length indexed type, its use cases, verifying its properties.	2					2	**4**	**5**	
3. User-defined datatypes.	2					2	**4**	**5**	
4. Type-driven development, programming with first-class and dependent types.	4					4	**8**	**6**	
5. Equality, rewrites in types, totality and decidability.	2					2	**4**	**4**	
6. Curry-Howard correspondence, proving by programming.	2					2	**4**	**3**	
7. Expressing complex subtypes with predicates.	2					2	**4**	**5**	
8. Decomposing data structures in alternative ways using views.	2					2	**4**	**5**	
9. Streams and infinite execution.	2					2	**4**	**5**	
10. Typed state machines as protocol specifications.	4					4	**8**	**6**	
11. Session types and concurrency.	2					2	**4**	**5**	
12. Linear types and resource management.	4					4	**8**	**6**	
13. Summary and relation to other programming languages.	2					2	**4**	**3**	
14. Preparing for the exam and taking the final exam (written).								**4**	
Total	**32**					**32**	**64**	**66**	

Assessment strategy	Weight,%	Deadline	Assessment criteria
Practical assignments	30	Each week.	Students perform small practical assignments related to the topic of the lecture. Any student should be able to present his work for a class, explaining how and why he came to the solution.
Lab assignment	30	Progress declared each week. Final presentation – 16th week of the semester.	Students choose a domain and develop a program by applying type-driven development practices. The program has to use all the main type-driven constructs like dependent types, typed state machines, session, and linear types, and others. Also, domain-specific properties have to be stated and proved. The student has to be able to explain all the decisions and re-implement a small function explaining all the steps. Progress has to be declared every week.
Exam (written)	40	Exam session.	Exam consists of open, semi-open and close-ended questions from the topics covered in lectures.

Author	Year	Title	Number or volume	Publisher or URL
Required reading				
Edwin Brady	2017	Type-Driven Development with Idris		Manning. ISBN: 978-1617293023
Recommended reading				
Adam Chlipala	2013	Certified Programming with Dependent Types		The MIT Press. ISBN: 978-0262026659
Samuel Mimram	2020	Program = Proof		Independently published. ISBN: 979-8615591839