COURSE UNIT DESCRIPTION

Course unit title	Course unit code
Software Engineering Methods and Tools	**PMKM7124**

Lecturer(s)	Department where the course unit is delivered
Coordinator: prof. dr. Romas Baronas **Other lecturers:** -	Department of Software Engineering Faculty of Mathematics and Informatics Vilnius University

Cycle	Level of course unit	Type of the course unit
Second	-	Compulsory

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

Prerequisites and corequisites	
Prerequisites: Requirements Engineering	**Corequisites (if any):** -

Number of ECTS credits allocated	Student's workload	Contact hours	Self-study hours
7	190	82	108

Purpose of the course unit: programme competences to be developed		
To increase knowledge of methods and tools of software requirements, design, construction, testing, maintenance, configuration and quality assessment, to increase expertise in the area of the conceptual and formal modeling, to develop skills in choosing and applying appropriate software engineering methods and tools in practice, to improve abstract thinking.		
Learning outcomes of the course unit: students will be able to	**Teaching and learning methods**	**Assessment methods**
Evaluate software engineering methods and tools and choose the right system for developing a specific application.	Lectures, problem-oriented teaching, case studies, information retrieval, literary reading, individual work, tutorials, laboratory work	Laboratory works and results presentation, written exam (open, semi-open and close-ended questions and tasks).
Formalize a subject area, to formally define the specific requirements and use them in the software development.		
Clearly present the chosen topic, summarize it, argue and defend his/her own opinion.	Information retrieval, literary readings, tutorials, report preparation and presentation at the seminar, group discussion, demonstration	Presentation material, the oral presentation, answers to oral questions

Course content: breakdown of the topics	Contact hours							Self-study work: time and assignments	
	Lectures	Tutorials	Seminars	Practice	Laboratory work	Practical training	**Contact hours**	**Self-study hours**	**Assignments**
1. The conception, purpose and classification of the software engineering methods and tools.	1						**1**	**1**	Self-study of literature.
2. Formal methods of software engineering, their classification, characteristics, application.	2						**2**	**2**	Self-study of literature.

3. Object constraint language (OCL) as an extension of UML. Kinds of constraints, structural and user-defined types, OCL expressions, OCL messages.	4	1			3		**8**	**12**	Self-preparation for 1st laboratory work: to create UML model of a chosen subject area (task), to define OCL requirements, and to generate program code using an appropriate tool. Self-study of literature.
4. An application of specific OCL constraints. Model-driven application development.	4		2		4		**10**	**12**	
5. Formal modeling with Z language. An implementation of models.	4	1	2		3		**10**	**16**	Self-preparation for 2nd laboratory work: to choose a formal software engineering method (Z system, algebraic specifications, program analysis-synthesis, etc.) as well as a tool supporting the method, and to apply them to a self-chosen software development task (problem). Self-study of literature.
6. Description of the system behavior by algebraic specifications.	4				2		**6**	**10**	
7. Prototyping the algebraic specifications.	2		2		2		**6**	**10**	
8. Formal program description by symbolic logic.	2						**2**	**5**	
9. Formal program analysis and synthesis by mechanical theorem proving.	3		2				**5**	**5**	
10. Software requirements, design, construction, testing, maintenance, configuration management and quality tools.	3	2	10				**15**	**15**	Preparation of a presentation on a modern software engineering method and/or a corresponding tool (tools), which is not used in the study program. Self-study of literature.
11. Software quality, engineering management and process modeling tools, case-tools, miscellaneous tools.	3	2	10				**15**	**15**	
12. Preparing for the exam and taking the final exam (written)							**2**	**5**	Self-study of literature. (5 hours - preparation for exam, 2 hours – exam).
Total	**32**	**6**	**28**		**14**		**82**	**108**	

Assessment strategy	**Weight, %**	**Deadline**	**Assessment criteria**
Oral presentation	30	During the semester	The following aspects of the presentation are assessed: - The presentation structure, size and style: the structure is clear and logical, contains all necessary components (introduction, explanation, conclusions), the presentation is of a reasonable duration; the material was delivered for a preview - 1 point. - Completeness, recommendations and conclusions: The material presented in detail and in comparison to others methods/tools, recommendations and conclusions are grounded - 2 points; if the material is incomplete or the given conclusions are unreasonable – not more than 1 point.
Two laboratory works	20	During the semester	The assessment of two laboratory works: 1. An application of OCL language in practice: work is fully (all the basic types of constrains were applied) completed and defended on time (within first 10 weeks) - 1.2 points. Unused kinds of constraints proportionately reduce the assessment. Lateness no more than 2 weeks leads to reducing the assessment in 25%, lateness no more than 4 weeks – 50%, later – 75%. 2. An application of another method and tool of software engineering: work is fully (all key elements of the method and the tool were meaningfully used) completed and defended – 0.8 points.
Exam (written)	50	Exam session	The exam consists of 25 open, semi-open and close-ended questions and tasks each of them is assessed between 0.1 and 0.3 points. The questions are formulated from topics set out in lectures. The exam is allowed only after reading the report of

			the seminar and the collection of at least 2 points before the exam. The assessment of the exam: - 5 points: excellent knowledge and skills, the assessment level, collected at least 4.5 points; - 4 points: good knowledge and skills, the synthesis level, collected at least 3.5 points; - 3 points: average knowledge and skills, the analysis level, collected at least 2.5 points; - 2 points: knowledge and skills are less than average, the application level, collected at least 1.5 points. - 1 point: knowledge and skills are too low, collected less than 1.5 points.

Author	Year	Title	Number or volume	Publisher or URL
Required reading				
J. Warmer, A. Kleppe	2003	The Object Constraint Language: Getting Your Models Ready for MDA	2nd ed.	Addison-Wesley Professional
H. Habrias, M. Frappier	2000	Software Specification Methods: An Overview Using a Case Study (Formal Approach to Computing and Information Technology)		Springer-Verlag
M. Christensen, M. Dorfman, R.H. Thayer (eds.).	2002	Software Engineering	Vol. 1, 2, 3rd ed.	Wiley-IEEE Computer Society Press
Recommended reading				
H. Hußmann	2002	Formal Specification of Software Systems		Technische Universität Dresden, http://www-st.inf.tu-dresden.de/fs/
A. Kleppe, J. Warmer, W. Bast	2003	MDA Explained: The Model Driven Architecture-Practice and Promise		Addison-Wesley Professional
S.L. Pfleeger	2009	Software Engineering: Theory and Practice	4th ed.	Prentice Hall
C.-L. Chang, R.C.-T. Lee	1997	Symbolic Logic and Mechanical Theorem Proving		Academic Press
OMG	2012	UML 2.3 OCL Specification		Object Management Group, http://www.omg.org/spec/OCL
J. Bowen	1996	Formal Specification and Documentation using Z: A Case Study Approach		Thomson Publishing http://www.zuser.org/zbook
J.L. Turner, T.L. McCluskey	1992	The Construction of Formal Specifications: An Introduction to the Model-Based and Algebraic Approaches		http://helios.hud.ac.uk/scomtlm/book.pdf