Software Engineering

	Prerequisites:	Computer Science Fundamentals, Programming Fundamentals (for lectures), Object-Oriented Programming			
	Learning outcomes:	As a result of discipline studying students should know: - software life-cycle models and phases - software development standards - basic concepts of BPMN and UML be able: - to analyze systems and specify system requirements - to design systems and its components - to create UML diagrams - to create technical documentation - to organize implementation, testing and deployment of software have an idea: - software quality management standards - software costs management - Model Driven Development - project patterns			
№	**Lecture**	**Hours**	**Laboratory works**		**Ref**
			Content	**Hours**	
1.	**Software engineering and its place as an engineering discipline**				
	Definitions, role of the Software Engineering; the costs of Software Engineering; Software Engineering Methods; CASE (Computer-Aided Software Engineering); the attributes of good software; the key challenges facing software engineering..	2 h			[2]
2.	**Software developing process**				
	Introduction to software life-cycle models and phases. Development process models as ways of organising activities: waterfall, V, evolutionary, Boehm's spiral model. Agile methods of software developing (XPprogramming). Choosing a development model. The main tasks undertaken by project managers; introduction to the software project management; project planning and the planning process.	2 h	Introduction to the Rational Unified Process model. The use of CASE tools to support project management and change management processes.	2 h	[1], [4]
3.	**Systems analyse**				
	BPM terminology. Business process modeling. The transition from the logical business service architecture to its implementation in software. Introduction to BPMN..	2 h	BPMN in practice - analyse of existing systems	2 h	[1], [4],[2]

4.	**Requirements engineering**				
	The concepts of user and system requirements; functional and non-functional requirements; requirements and design; eliciting requirements. Requirements specification and technical documentation creation process. Specification languages; IEEE requirements standards.	2 h	Practical tutorials on how software requirements may be organised in a requirements document. Use Case Diagrams and Sequence Diagrams. Developing a simple set of requirements (to be done as a team) for some innovative client/server applications, using the results of analysis and BPMN diagrams	2 h	[2],[3]
5.	**System modeling (maybe it is should be number 4 and nr 4 labs should extend labs from number 3 -> to disscuss)**				
	Modeling principles; the idea of modelling: data, functionality, data flow, control, consistency of models; pre & post conditions, invariants; introduction to mathematical models and specification languages; types of models. Applying UML notation to system modelling. Project patterns.	2 h	Basic Modeling Skills in UML: class diagrams and activity charts.	2 h	[2] [3]
6.	**Architectural design**				
	Introduction to architectural design and its importance; the architectural design decisions; three complementary architectural styles covering organisation, decomposition and control; reference architectures that are used to communicate and compare architectures.	2 h	Familiarization with the distributed architectures, pipe-and-filter, model-view-controller.	2 h	[1],[2]
7.	**Model-based techniques in Validation & Verification**				
	The idea of correctness; difference between validation and verification; V&V by Inspection; V&V by Testing; V&V by automated analysis (static analysis, model checking and proof).	2 h	Study and writing models for small applications and its proof using OCL.	2 h	[1], [2],[3]
8.	**Special issues**				
	Delivery and postdelivery phases. SQM. SCM. Integrating SQM into the development lifecycle. Software quality. Software development standards. Understanding Software Quality Assurance, Software Quality Plan (SQP) and Software Quality Control (SQC). Software measurement and metrics.	2 h	Study and writing applications using aspect-oriented programming principles and introduction to open source development	2 h	[2]
	TOTAL	**16 h**		**14 h**	

References

1. I Sommerville Software Engineering (8th Edition)
2. R. Pressman Software Engineering: A Practitioner's Approach (6th Edn), Published by McGraw-Hill Science/Engineering/Math, 2004, 880p
3. D Bell Software Engineering: A Programming Approach Published by Addison Wesley, 2000, 488p.
4. M Fowler UML Distilled - a brief guide to the standard object modeling language (3rd Edition) Published by Addison-Wesley Professional, 185p.