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	**Master**
1.6 Study programme / Qualification	**Software Engineering**

2. Information regarding the discipline

2.1 Name of the discipline (en) (ro)	 **Methodologies for Software Processes** **Metodologii pentru Procese Software**
2.2 Course coordinator	**Assoc. Prof. Eng. Florin Craciun**
2.3 Seminar coordinator	**Assoc. Prof. Eng. Florin Craciun**

2.4. Year of study	**1**	2.5 Semester	**2**	2.6. Type of evaluation	**E**	2.7 Type of discipline	**DF**

3. Total estimated time (hours/semester of didactic activities)

3.1 Hours per week	4	Of which: 3.2 course	2	3.3 seminar/laboratory	2
3.4 Total hours in the curriculum	56	Of which: 3.5 course	28	3.6 seminar/laboratory	28

Time allotment:	hours
Learning using manual, course support, bibliography, course notes	30
Additional documentation (in libraries, on electronic platforms, field documentation)	10
Preparation for seminars/labs, homework, papers, portfolios and essays	59
Tutorship	10
Evaluations	10
Other activities:	-

3.7 Total individual study hours	119
3.8 Total hours per semester	175
3.9 Number of ECTS credits	7

4. Prerequisites (if necessary)

4.1. curriculum	• None
4.2. competencies	• Basic software development skills

5. Conditions (if necessary)

5.1. for the course	

6. Specific competencies acquired

Professional competencies	• Understanding and working with basic concepts in software engineering; • Capability of analysis and synthesis; • Proficient use of methodologies and tools specific tool software systems • Organization of software production processes.
Transversal competencies	• Team work capabilities; able to fulfill different roles • Professional communication skills; concise and precise description, both oral and written, of professional results, • Antepreneurial skills;

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

7.1 General objective of the discipline	• be able to apply basic methods for software process formalization
7.2 Specific objective of the discipline	• know the main features of the common software process models. • be able to represent the software processes using SPEM standard. • be able to create new software processes. • be able to use CASE tools for authoring, configuring and publishing software processes • know the principles of different software development methodologies: model driven development, agile model driven development, feature driven development, use case driven development, domain driven development, test driven development.

8. Content

8.1 Course	Teaching methods	Remarks
1. Software Process Concepts. Definitions. Main concepts: role, work product, activity.	Exposure,description, explanation, debate and dialogue, discussion of case studies	
2. Software Process Models. Typical tasks and life cycle of the more common software development models: ad-hoc development, waterfall model, v-model, iterative development, prototyping, rapid application development, exploratory model, spiral model, reuse model, unified process.	explanation, debate and dialogue, discussion of case studies	

3. Software and System Process Engineering Meta-Model (SPEM). Meta-model architecture and principles. SPEM UML profile. Core. Process structure. Process behavior.	Exposure,description, explanation	
4. Software and System Process Engineering Meta-Model (SPEM). Managed content. Method content. Process with methods. Method Plugin. Process diagrams.	Exposure,description, explanation	
5. Software Process Frameworks. Eclipse Process Framework Project (EPF).	Exposure,description, explanation, discussion of case studies	
6. Software Process Frameworks. Eclipse Open Unified Process (OpenUP).	Exposure,description, explanation, discussion of case studies	
7. Model Driven Architecture (MDA). Basic Concepts. MDA transformations.	Exposure,description, explanation,	
8. Model Driven Architecture (MDA). Query/View transformation (QVT). Model to text transformation (M2T).	Exposure,description, explanation	
9. Agile Model Driven Development (AMDD). Agile modeling. Principles. Best practices. Approaches for applying AMDD on projects.	Exposure,description, explanation, discussion of case studies	
10. Feature Driven Development (FDD). FDD process. Feature oriented software development (FOSD). FOSD phases. Software product lines.	Exposure,description, explanation, discussion of case studies	
11. Use Case Driven Development. Goal driven view. Types of alternative courses. Use case fundamentals.	Exposure,description, explanation, discussion of case studies	
12. Use Case Driven Development. Practical issues. Iconix process.	Exposure,description, explanation, discussion of case studies	
13. Domain Driven Development (DDD). Ubiquitous language. Bounded contexts. Layered architecture. Aggregates. Factories. Repositories. Services.	Exposure,description, explanation, discussion of case studies	
14. Test Driven Development (TDD). Fundamentals. Examples.	Exposure,description, explanation, discussion of case studies	

Bibliography

1. Steve Adolph, Paul Bramble, Alistair Cockburn, and Andy Pols, Patterns for Effective Use Cases, Addison-Wesley, 2002.

2. Scott W. Ambler, Agile Model Driven Development: The Key to Scaling Agile Software Development, 2009, http://www.agilemodeling.com/essays/amdd.htm

3. Sven Apel and Christian Kastner, An overview of Feature-Oriented Software Development,

Journal of Object Technology, vol. 8, no. 5, July-August 2009.

4. Kent Beck, Test-Driven Development by Example, Addison-Wesley, 2002.
5. Eric Evans, Domain-Driven Design, Addison-Wesley, 2004.
6. Eclipse Process Framework Project (EPF), 2010, http://www.eclipse.org/epf/
7. Eclipse Open Unified Process (OpenUP), 2010, http://epf.eclipse.org/wikis/openup
8. OMG, Model-Driven Architecture, 2003, http://www.omg.org/cgi-bin/doc?omg/03-06-01
9. OMG, Software & Systems Process Engineering Meta-Model Specification (SPEM) version 2.0, 2008, http://www.omg.org/spec/SPEM/2.0/
10. Clay Williams, Matthew Kaplan, Tim Klinger, and Amit Paradkar, Toward Engineered, Useful Use Cases, Journal of Object Technology, vol. 4, no. 6, August 2005.

8.2 Seminar / laboratory	Teaching methods	Remarks
1. Project1: Describe a software process in SPEM	Use practical tools to implement group projects. Discuss research papers.	Seminar is organized as a total of 14 hours – 2 hours every second week Project is organized as a total of 14 hours – 2 hours every
2. Seminar1: Discuss research papers	Use practical tools to implement group projects. Discuss research papers.	
3. Project2: Describe a software process in SPEM	Use practical tools to implement group projects. Discuss research papers.	
4. Seminar2: Discuss research papers	Use practical tools to implement group projects. Discuss research papers.	
5. Project3: Describe a software process in SPEM	Use practical tools to implement group projects. Discuss research papers.	
6. Seminar3: Discuss research papers	Use practical tools to implement group projects. Discuss research papers.	
7. Project4: A project in EPF	Use practical tools to implement group projects. Discuss research papers.	
8. Seminar4: Discuss research papers	Use practical tools to implement group projects. Discuss	

	research papers.	
9. Project5: A project in EPF	Use practical tools to implement group projects. Discuss research papers.	
10. Seminar5: Discuss research papers	Use practical tools to implement group projects. Discuss research papers.	
11. Project6: A project in EPF	Use practical tools to implement group projects. Discuss research papers.	
12. Seminar6: Discuss research papers	Use practical tools to implement group projects. Discuss research papers.	
13. Project7: A project in EPF	Use practical tools to implement group projects. Discuss research papers.	
14. Seminar7: Discuss research papers	Use practical tools to implement group projects. Discuss research papers.	
Bibliography two practical tools: MagicDraw and EPF. Research papers		

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 Software Engineering studies;
- The content of the course is considered by the software companies as important for average software development skills

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	Written exam	40.00%
10.5 Seminar/lab activities - e -	- be able to implement course concepts – be able to use tools for different software process concept - be able to do a critical evaluation of research papers - to be able to write a critical	-Practical examination	60.00%

	essay		
10.6 Minimum performance standards			
➢ At least grade 5 (from a scale of 1 to 10) at both written exam and laboratory work.			

Date Signature of course coordinator Signature of seminar coordinator

.... Assoc. Prof. En. Florin CRACIUN Assoc. Prof. Eng. Florin CRACIUN

Date of approval Signature of the head of department

....…..........................