SYLLABUS

1. Information regarding the programme

1.1 Higher education institution	**Babes 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	**Bachelor**
1.6 Study programme / Qualification	**Computer Science**

2. Information regarding the discipline

2.1 Name of the discipline	**Software engineering**						
2.2 Course coordinator	**conf. dr. Dan CHIOREAN**						
2.3 Seminar coordinator	**asist. drd. Dragos PETRASCU**						
2.4. Year of study	**2**	2.5 Semester	**4**	2.6. Type of evaluation	**E**	2.7 Type of discipline	**Compulsory**

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	1S + 1L
3.4 Total hours in the curriculum	56	Of which: 3.5 course	28	3.6 seminar/laboratory	1/1
Time allotment:					hours
Learning using manual, course support, bibliography, course notes					27
Additional documentation (in libraries, on electronic platforms, field documentation)					14
Preparation for seminars/labs, homework, papers, portfolios and essays					23
Tutorship					10
Evaluations					20
Other activities:					

3.7 Total individual study hours	94
3.8 Total hours per semester	150
3.9 Number of ECTS credits	6

4. Prerequisites (if necessary)

4.1. curriculum	• Object-Oriented Programming
4.2. competencies	• Average programming skills in a high level object-oriented programming language

5. Conditions (if necessary)

5.1. for the course	• beamer
5.2. for the seminar /lab activities	• Laboratory with computers; high level programming language environment (Java environments or .NET and a UML CASE tool)

6. Specific competencies acquired

Professional competencies	• C2.1 & C2.2 - Knowledge on modeling, software development metodologies, software testing, project management • C2.3 - Ability to work independently and in a team in order to develop software complying with industrial standards. • C2.5 - Understanding the role of different artifacts used in the process of software development and acquiring the ability of realizing and using these artifacts
Transversal competencies	• CT1 - Ability to create different models (analysis, design, implementation, testing) using the UML • CT2 - Ability to create software beginning with model construction, continuing with model verification and model transformation in code, realizing and using testing models • CT3 - Ability to use a software methodology to produce quality software from analysing software requirements to code generation and software testing

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

7.1 General objective of the discipline	• Be able to understand software production life cycle • Improved skills on developing software
7.2 Specific objective of the discipline	• Understand and work with the concepts of: model, model properties. Understanding the role of abstraction in producing models. • Understand the differences between modeling languages and modeling methodologies. • Understand and work with the most important UML concepts used in constructing software models

8. Content

8.1 Course	Teaching methods	Remarks
1. Introduction to Software Engineering	Exposure: description, explanation, examples, discussion of case studies	
2. Using UML to specify models	Exposure: description, explanation, examples, discussion of case studies	
3. Requirements Elicitation	Exposure: description, explanation, examples, discussion of case studies	
4. Analysis	Exposure: description, explanation, examples, discussion of case studies	
5. System Design - Decomposing the System	Exposure: description, explanation, examples, discussion of case studies	
6. System Design - Addressing Design Goals	Exposure: description, explanation, examples, discussion of case	

	studies	
7. Object Design - Reusing Pattern Solutions	Exposure: description, explanation, examples, discussion of case studies	
8. Object Design - Specifying Interfaces	Exposure: description, explanation, examples, discussion of case studies	
9. Mapping Models to Code	Exposure: description, explanation, examples, discussion of case studies	
10. Testing	Exposure: description, explanation, examples, discussion of case studies	
11. Rationale & Configuration Management	Exposure: description, explanation, examples, discussion of case studies	
12. Project Management	Exposure: description, explanation, examples, discussion of case studies	
13. Software Life Cycle	Exposure: description, explanation, examples, discussion of case studies	
14. Methodologies	Exposure: description, explanation, examples, discussion of case studies	

Bibliography

1. Bernd Bruegge, Allen Dutoit - Object-Oriented Software Engineering Using UML, Patterns and Java - 3rd Edition - Prentice Hall 2009
2. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides - Design Patterns - Addison-Wesley, 1996
3. Ian Sommerville - Software Engineering - 8th edition - Addison-Wesley, 2006
4. Grady Booch, James Rumbaugh, Ivar Jacobson - The Unified Modeling Language User Guide, V.2.0 - Addison Wesley, 2005
5. Martin Fowler et al. - Refactoring - Improving the Design of Existing Code - Addison Wesley, 1999

8.2 Seminar	Teaching methods	Remarks
1. Use cases diagrams, concepts, relationships, representation, the structure of a use case description document	Explanation, Dialogue, debate, case studies, examples, proofs	The seminar is structured as 2 hours classes at each two weeks period
2. Describing structural models using class diagrams - concepts, relationships, representation, filtering the information	Explanation, Dialogue, debate, case studies, examples, proofs	
3. Describing behavioural models using sequence and collaboration diagrams - the concepts used in these diagrams, the equivalence of these diagrams	Explanation, Dialogue, debate, case studies, examples, proofs	

4. Describing behavioural models using state transition diagrams. Generating code from state class diagrams	Explanation, Dialogue, debate, case studies, examples, proofs	
5. Using assertions to specify model correctness against different kind of rules. Code generation for UML models	Explanation, Dialogue, debate, case studies, examples, proofs	
6. The role of pre-post-conditions in specifying component's interface - design by contract	Explanation, Dialogue, debate, case studies, examples, proofs	
7. Testing patterns	Explanation, Dialogue, debate, case studies, examples, proofs	

Bibliography

1. Martin Fowler - UML Distilled - Addison-Wesley, 2003
2. Bruce Eckel - Thinking in Java 4th edition - Prentice Hall, 2006
3. Kent Beck - Test Driven Development - Addison-Wesley, 2002

8.2 Laboratory	Teaching methods	Remarks
1. Agile Software Methodologies - planning the software development phases. Risk analysis in software development, the role of incremental and iterative development. Analysis of small software applications that each student has to analyse, design, implement and test.	Explanation, dialogue, case studies	The laboratory is structured as 2 hours classes at each two weeks period
2. Using an UML CASE tool and text editors to realize the functional model of each individual problem	Explanation, dialogue, case studies	
3. Using an UML CASE tool to construct The requirement model of each individual problem	Explanation, dialogue, case studies	
4. Constructing the Design model using an UML CASE tool	Explanation, dialogue, case studies	
5. Realizing the Implementation model using both an UML CASE tool and an appropriate IDE	Testing data discussion, evaluation	
6. Testing the application realized	Testing data discussion, evaluation	
7. Realizing the User manual and delivering the application	Explanation, dialogue, case studies	

Bibliography

1. Kenneth S. Rubin - Essential Scrum - A Practical Guide to the Most Popular Agile Process - Addison-Wesley 2012
2. Philippe B. Kruchten - The Rational Unified Process: An Introduction - 3rd Edition Addison - Wesley 2003
3. Per Kroll, Philippe Kruchten and Grady Booch - The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP - Addison-Wesley 2003

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 Curricula Recommendations for Computer Science Studies;
- The course exists in the studying program of all major universities in Romania and abroad;
- The content of the course contains knowledge mandatory for any IT specialist working in a software company

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 concepts & SE principles; - knowledge of UML key concepts - knowledge of modeling methodologies	Written exam	60%
10.5 Seminar/lab activities	- be able to implement acknowledged knowledge in producing software - be able to produce and use modeling artifacts	- Practical examination - documentation -continuous observations	40%

Date	Signature of course coordinator	Signature of seminar coordinator
11 May 2014	conf. dr. Dan CHIOREAN	asist. drd. Dragos PETRASCU

Date of approval

Signature of the head of department

prof. dr. Bazil PÂRV