**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 | **Computers and Information Technology** |
| 1.5 Study cycle | **Bachelor** |
| 1.6 Study programme / Qualification | **Information Engineering** |

**2. Information regarding the discipline**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 2.1 Name of the discipline (en)<br>(ro) | | | Software Engineering<br>Inginerie software | | | | |
| 2.2 Course coordinator | | | Lect. dr. Vladiela Petrașcu | | | | |
| 2.3 Seminar coordinator | | | Lect. dr. Vladiela Petrașcu | | | | |
| 2.4. Year of study | 3 | 2.5 Semeste | 6 | 2.6. Type of evaluation | E | 2.7 Type of discipline | Compulsory DD |
| 2.8 Code of the discipline | | MLE5177 | | | | | |

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

| | | | | | |
|---|---|---|---|---|---|
| 3.1 Hours per week | 6 | Of which: 3.2 course | 2 | 3.3 seminar/laboratory | 1 S<br>1 LP<br>1 P |
| 3.4 Total hours in the curriculum | 70 | Of which: 3.5 course | 28 | 3.6 seminar/laboratory | 42 |
| Time allotment: | | | | | Hours |
| Learning using manual, course support, bibliography, course notes | | | | | 19 |
| Additional documentation (in libraries, on electronic platforms, field documentation) | | | | | 18 |
| Preparation for seminars/labs, homework, papers, portfolios and essays | | | | | 28 |
| Tutorship | | | | | 7 |
| Evaluations | | | | | 8 |
| Other activities: ..... | | | | | |

| | |
|---|---|
| 3.7 Total individual study hours | 80 |
| 3.8 Total hours per semester | 150 |
| 3.9 Number of ECTS credits | 6 |

**4. Prerequisites** (if necessary)

| | |
|---|---|
| 4.1. curriculum | • Fundamentals of Programming<br>• Object-Oriented Programming |
| 4.2. competencies | • Programming in a high-level object-oriented language |

**5. Conditions** (if necessary)

| | |
|---|---|
| 5.1. for the course | • Videoprojector |
| 5.2. for the seminar /lab activities | • Computers<br>• UML Case Tool<br>• Java/.NET IDE |

**6. Specific competencies acquired**

| | |
|---|---|
| **Professional competencies** | C4.1 Identifying and describing technologies, programming environments and various concepts that are specific to programming engineering<br><br>C4.2 Explaining the role, interaction and operation patterns of software system components<br><br>C4.3 Developing specifications and designing information systems using specific methods and tools<br><br>C4.4 Managing the life cycle of hardware, software and communications systems based on performance evaluation<br><br>C4.5 Developing, implementing and integrating software solutions |
| **Transversal competencies** | CT1 Honorable, responsible, ethical behavior, in the spirit of the law, to ensure the professional reputation<br>CT2 Identifying, describing and conducting processes in the project management field, undertaking different team roles and clearly and concisely describing own professional results, verbally or in writing.<br>CT3 Demonstrating initiative and pro-active behavior for updating professional, economical and organizational culture knowledge |

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

| | |
|---|---|
| 7.1 General objective of the discipline | • Aquiring knowledge of and applying sound concepts, principles and engineering techniques when building software systems |
| 7.2 Specific objective of the discipline | • Aquiring knowledge of software lifecycle stages and process models<br>• Understanding software modeling<br>• Aquiring knowledge of and applying model-based software development techniques<br>• Getting used to correctly apply the UML language<br>• Aquiring ability to use UML Case tools<br>• Aquiring basic project management knowledge<br>• Aquiring knowledge of software development methodologies, both traditional and agile |

**8. Content**

| 8.1 Course | Teaching methods | Remarks |
|---|---|---|
| 1. Introduction to Software Engineering: motivation, definitions, concepts, activities | Explanation, conversation, discussing case studies | |
| 2. Software lifecycle stages. Software process models | Explanation, conversation, discussing case studies | |
| 3. Software complexity management techniques (abstraction, decomposition, modeling). Modeling in Software Engineering: definitions, model types and modeling tools | Explanation, conversation, discussing case studies | |
| 4. Introduction to the UML language: concepts, diagram types, syntax/semantics, tools | Explanation, conversation, discussing case studies | |
| 5. Requirements Elicitation: concepts, activities, examples | Explanation, conversation, discussing case studies | |
| 6. Requiements Analysis: concepts, activities, examples | Explanation, conversation, discussing case studies | |
| 7. System Design: concepts, principles, activities | Explanation, conversation, discussing case studies | |
| 8. Object Design: concepts, principles, activities | Explanation, conversation, discussing case studies | |
| 9. Object Design - Design Patterns | Explanation, conversation, discussing case studies | |
| 10. Object Design – Interface Specification. Design by Contract – using assertions in modeling | Explanation, conversation, discussing case studies | |
| 11. System Implementation. Model-based code generation: concepts, principles, activities, examples | Explanation, conversation, discussing case studies | |
| 12. Software Verification and Validation | Explanation, conversation, discussing case studies | |
| 13. Software Management: concepts and activities | Explanation, conversation, discussing case studies | |
| 14. Software Development Methodologies. Model Driven Engineering (MDE) | Explanation, conversation, discussing case studies | |

Bibliography

Bibliografie

[1] Booch, G., Rumbaugh, J., Jacobson, I., *The Unified Modeling Language User Guide - V.2.0,* Addison Wesley, 2005.

**[2] Brambilla, M., Cabot, J., Wimmer, M., *Model-Driven Software Engineering in practice* – 2nd edition, Morgan and Claypool Publishers, 2017.**

**[3] Bruegge, B., Dutoit, A., *Object-Oriented Software Engineering Using UML, Patterns and Java – 3rd ed.*, Pearson Education, 2014.**

[4] Fowler, M. et al., *Refactoring - Improving the Design of Existing Code*, Addison Wesley, 1999.

[5] Fowler, M, *UML Distilled: A Brief Guide to the Standard Object Modeling Language* - 3*rd ed*., Addison-Wesley, 2003.

[6] Gamma, E., Helm, R., Johnson, R., Vlissides, J., *Design Patterns*, Addison-Wesley, 1996.

[7] Martin, R.C., *Agile Software Development: Principles, Patterns, and Practices*, Prentice Hall, 2002.

[8] Pressman, R.S., *Software Engineering - A Practitioners Approach - 8th ed.*, McGraw-Hill, 2014.

**[9] Seidl, M., Scholz, M., Huemer, C., Kappel, G., *UML @ Classroom: An Introduction to Object-Oriented Modeling*, Springer International Publishing, 2015.**

[10] Schach, S.R., *Object-Oriented and Classical Software Engineering - 8th ed.*, McGraw-Hill, 2010.

[11] Sommerville, I., *Software Engineering - 10th ed.*, Pearson, 2015.

Links:

[1] OMG UML 2.5.1 - About the Unified Modeling Language Specification Version 2.5.1 (omg.org)

[2] OMG OCL 2.4 - About the Object Constraint Language Specification Version 2.4 (omg.org)

[3] StarUML - StarUML

[4] OCLE - OCLE 2.0 - Object Constraint Language Environment (ubbcluj.ro)

[5] Eclipse Modeling Framework - Eclipse Modeling Project | The Eclipse Foundation

| **8.2 Seminar** | **Teaching methods** | **Remarks** |
|---|---|---|
| 1. Using Use Case Diagrams to describe a functional model: concepts, relations, syntax, use case description templates | explanation, conversation, arguing, exemplifying | A 2h seminar every other week |
| 2. Using Class Diagrams to describe structural models: concepts, relations, syntax, problem domain model vs. solution model | explanation, conversation, arguing, exemplifying | |
| 3. Using Sequence/Communication Diagrams to describe dynamic models: concepts. syntax, equivalence | explanation, conversation, arguing, exemplifying | |
| 4. Using Statechart Diagrams to describe dynamic models. The *State* Design Pattern | explanation, conversation, arguing, exemplifying | |
| 5. The use of assertions in modeling. Design by Contract | explanation, conversation, arguing, exemplifying | |
| 6. Automatic code generation based on UML/OCL models | explanation, conversation, arguing, exemplifying | |

| | | |
|---|---|---|
| 7. Testing: concepts, principles, tools | explanation, conversation, arguing, exemplifying | |

| **8.3 Laboratory** | **Teaching methods** | **Remarks** |
|---|---|---|
| 1. Agile methodologies: planning software development. Investigating various UML/OCL Case Tools (ex. StarUML, OCLE) | explaining, arguing, exemplifying | A 2h lab every other week |
| 2. Using an UML Case Tool for drawing Use Case Diagrams | explaining, arguing, exemplifying | |
| 3. Using an UML Case Tool for drawing Class Diagrams corresponding to the problem domain | explaining, arguing, exemplifying | |
| 4. Using an UML Case Tool for drawing Sequence/Communication Diagrams and refining the structural model | explaining, arguing, exemplifying | |
| 5. Using an UML Case Tool for drawing Statechart Diagrams | explaining, arguing, exemplifying | |
| 6. Using an UML/OCL Case Tool for specifying/evaluating assertions on UML models | explaining, arguing, exemplifying | |
| 7. Using an UML/OCL Case Tool for code generation | explaining, arguing, exemplifying | |

| **8.4 Project** | **Teaching methods** | **Remarks** |
|---|---|---|
| 1. Assigning to each student a small/medium size application that he/she should build, passing through all lifecycle stages and developing the corresponding models | Arguing, exemplifying | A 2h lab every other week |
| 2. Requirements Elicitation: using an UML Case tool and a text editor for developing the functional model of the application. 3 iterations use case planning. Developing a GUI prototype | Arguing, exemplifying | |
| 3. Requirements Analysis: using an UML Case tool for developing the domain (conceptual) model | Arguing, exemplifying | |
| 4. Software Design & Implementation: using an UML Case tool for developing dynamic models (interaction diagrams) and an IDE for implementing the use cases corresponding to the 1st iteration | Arguing, exemplifying | |
| 5. Software Design & Implementation: using an UML Case tool for developing dynamic models (interaction diagrams) and an IDE for implementing the use cases corresponding to the 2nd & 3rd iterations | Arguing, exemplifying | |
| 6. Testing the application | Arguing, exemplifying | |
| 7. Creating the user guide and delivering the buit application | Arguing, exemplifying | |

**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**

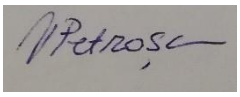
- The course obeys to the ACM/IEEE curricula guidelines for computer science study programs
- Similar courses are taught at most universities in Romania having similar study programs
- Software companies view this course as offering important background knowledge for future software developers

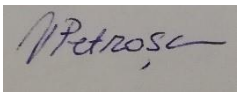
**10. Evaluation**

| Type of activity | 10.1 Evaluation criteria | 10.2 Evaluation methods | 10.3 Share in the grade (%) |
|---|---|---|---|
| 10.4 Course/Seminar | Knowledge of the basic software engineering concepts and principles taught | Written exam | 60% |
| | Software modeling knowledge and ability to use the UML language in this purpose | | |
| 10.5 Laboratory/Project | Applying aquired knowledge in building a small/medium-sized software system | Project | 40% |
| 10.6 Minimum performance standards | | | |
| ➢ At least grade 5 at both written exam and project | | | |

Date | Signature of course coordinator | Signature of seminar coordinator

17.05.2022





Date of approval | Signature of the head of department

Prof. dr. Laura Dioşan

24.05.2022