Advanced Concepts in Software Engineering

1. Information about the program

1.1 Higher education institution	University POLITEHNICA of Bucharest
1.2 Faculty	Faculty of Engineering in Foreign Languages
1.3 Department	Department of Engineering in Foreign Languages
1.4 Field of study	Computers and Information Technology
1.5 Study cycle	Master
1.6 Program / Qualification	Software Engineering

2. Data about the subject

2.1 Name of subject			Advanced Concepts in Software Engineering				
2.2 Course holder							
2.3 Seminar holder							
2.4 Laboratory/project holder							
2.5 Year of study	1	2.6 Semester	1	2.7 Evaluation type	E	2.8 Subject type	DPA/DO

3. Estimated time (hours per semester) of didactic activities

3.1 Number of hours per week	3	course hours	2	seminar		laboratory	1
3.2. Number of hours per semester	42	course hours	28	seminar		laboratory	14
3.3.Distribution of spend time:							**h.**
Study of textbooks, bibliography and course notes						24	
Supplementary study in library, on electronic platforms, on the fieldwork						12	
Preparation of seminars/laboratories, home assignments, papers, portfolios, essays						20	
Tutoring						5	
Examinations						5	
Other activities							

3.4 Total hours of individual study	**66**	
3.5 Total hours per semester	**108**	
3.6 Number of credits	**4**	

4. Preconditions (where relevant)

4.1 curriculum- related	• Introduction to Information Technology • Programming Languages • Data Structures and Algorithms
4.2 competence - related	• Programing skills

5. Facilities and equipment (where relevant)

5.1 for the course	• Overhead Projector
5.2 for the course seminar	•
5.3 for the laboratory/project	• 20 PC

6. Specific competences acquired

Professional competences	• Design appropriate software solutions, using responsible engineering approaches and applying theories and models that provide a basis for software design • Work effectively in interdisciplinary contexts, in particular to bridge the gap between computing technology and the clients business and to interpret and respect extra-technical constraints deriving from the business organization
	• Understand and be able to use specific tools, components, and frameworks and also abstract elements such as algorithms and architectures • Organize and lead development teams, including team-building and negotiation • Serve as an agent of change for introducing new technology

7. Course objectives (as resulting from the grid of specific competences)

7.1 Subject general goal	• The aim of this course is to introduce the issues, basic and advanced principles of software engineering. The objectives are to develop a framework into which more detailed material regarding specific aspects of the software engineering process techniques and issues can fit, including requirements, verification, testing, validation and quality processes.
7.2 Specific objectives	• Introduction of students to main notions of Software Engineering • Develop the competencies to manage the software development process • Develop the competencies to develop documentation, work in team , develop a specific product in a sound software engineering manner

8. Content

Course	Teaching methods	Observations
Software Processes 1.1. Software process models 1.2. Process iteration 1.3. Process activities 1.4. Computer-aided software engineering 1.5. CMM approach	Lecturing	
Requirements Engineering Processes 2.1. Feasibility studies 2.2. Requirements elicitation and analysis 2.3. Requirements validation 2.4. CASE tools for requirements management 2.5. Critical system specification 2.6. Formal specification	Lecturing	
Verification and Validation 3.1. Planning verification and validation 3.2. Software inspections 3.3. Automated static analysis 3.4. System and component testing 3.5. Test case design 3.6. CASE tools for testing automation	Lecturing	
Configuration Management 4.1. Configuration management planning 4.2. Change management 4.3. Version and release management 4.4. CASE tools for configuration management	Lecturing	
Software Cost Estimation 5.1. Software productivity 5.2. Estimation techniques 5.3. Project duration and staffing	Lecturing	
Quality Management 6.1. Process and product quality 6.2. Quality assurance and standards 6.3. Quality planning 6.4. Software measurements and metrics	Lecturing	

Bibliography

- Sommerville, I, "Software Engineering 7", 7th Ed., Addison Wesley, 2004
- Hughes, B and Cotterell, M, "Software Project Management", 3rd Ed, McGraw Hill, 2002
- Humphrey, W, "Managing the Software Process", SET Series in Software Engineering, Addison Wesley 1990
- Boehm, B, "Software Engineering Economics", Prentice Hall, 1981
- Kan, Stephen "Metrics and Models in Software Quality Engineering" 2nd Ed, Addison Wesley Professional, 2002
- Blanchard, B. S and Fabrycky W.J, , Systems Engineering and Analysis, Prentice Hall, NJ, 1997
- Smith, C.U.: Performance Engineering of Software Systems. Addison Wesley, 1990

8.3 Laboratory		
Requirements specification and management	Laboratory Teaching	
Architectural design	Laboratory Teaching	
Software design	Laboratory Teaching	
Software Testing	Laboratory Teaching	
Configuration Management	Laboratory Teaching	
Software Validation	Laboratory Teaching	
A comprehensive software application is studied and developed with automated tools	Project Tutoring	

Bibliography

- Hetzel, B, "The Complete Guide To Software Testing", 2nd Ed, QED Information Sciences, Inc, 1988
- SEI, SE-CMM, The Systems Engineering CMM, Architecture papers

9. Subject's relevance to the epistemic community, professional associations and representative employers in fields significant for the program

- **The software engineers adopt a systemstic and organised approach to their work, as this is often the most effective way to produce high-quality software. In this course the students are educated in advanced software engineering concepts that covers the entire development chain, from the business management perspective to the technical management and the development perspectives.**

10. Assessment

Activity	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Weight in final grade
10.4 Course	Course Presence and Activities	Presence and Activities Evaluation	10%
	Final Examinations	Written Exam	40%
10.5 Seminar			
10.6 Laboratory/Project	Laboratory Assignment	Assignments Correction	20%
	Project	Project Evaluation	30%
10.7 Minimal standard of performance			
• **Minimum 50% of the marks from Laboratory/Project and Course Activities part (3 points out of 6)** • **Minimum of 50 % from final examination (2 points out of 4)**			

Game and Interactive Simulation Systems

1. Information about the program

1.1 Higher education institution	University POLITEHNICA of Bucharest
1.2 Faculty	Faculty of Engineering in Foreign Languages
1.3 Department	Department of Engineering in Foreign Languages
1.4 Field of study	Computers and Information Technology
1.5 Study cycle	Master
1.6 Program / Qualification	Software Engineering

2. Data about the subject

2.1 Name of subject			Game and Interactive Simulation Systems				
2.2 Course holder							
2.3 Seminar holder							
2.4 Laboratory/project holder							
2.5 Year of study	1	2.6 Semester	1	2.7 Evaluation type	E	2.8 Subject type	DAP/DO

3. Estimated time (hours per semester) of didactic activities

3.1 Number of hours per week	3	course hours	1	seminar		laboratory	2
3.2. Number of hours per semester	42	course hours	14	seminar		laboratory	28
3.3.Distribution of spend time:							**h.**
Study of textbooks, bibliography and course notes							14
Supplementary study in library, on electronic platforms, on the fieldwork							
Preparation of seminars/laboratories, home assignments, papers, portfolios, essays							42
Tutoring							
Examinations							4
Other activities							

3.4 Total hours of individual study	**60**	
3.5 Total hours per semester	**102**	
3.6 Number of credits	**4**	

4. Preconditions (where relevant)

4.1 curriculum- related	•
4.2 competence - related	• Programming languages

5. Facilities and equipment (where relevant)

5.1 for the course	• Projector
5.2 for the course seminar	•
5.3 for the laboratory/project	• Computers with performant graphic boards, human-computer interfaces, 3D glasses, VR devices

6. Specific competences acquired

Professional competences	• Programming languages awareness for effective programming, including code, components and services creation, and integration of multiple subsystems
Transversal competences	• Understand and be able to use specific tools, components, and frameworks and also abstract elements such as algorithms and architectures • Organize and lead development teams, including team-building and negotiation • Serve as an agent of change for introducing new technology

7. Course objectives (as resulting from the grid of specific competences)

7.1 Subject general goal	• The course teaches students knowledge and skills necessary to develop games, interactive and Virtual Reality applications.
7.2 Specific objectives	• The course aims to introduce: -Familiarity with basic techniques for game design (sprites, 3D models, interactivity, reaction) -Usage of GameMaker Studio and Unity 5.x -Create 3D and VR models - Usage of HCI.

8. Content

8.1 Course	Teaching methods	Observations

Introduction, History, Classification.	slides	
GameMaker Studio, GML	slides	
Unity 5.x Interface	slides	
Unity 5.x 2D Applications	slides	
Unity 5.x 3D Applications	slides	
Human-Computer Interaction	slides	
Virtual Reality	slides	

Bibliography

1. Mr Ben G Tyers, Practical GameMaker: Studio: Language Projects, Apress, 2016
2. Nathan Auckett, GameMaker Essentials, Packt Publishing, 2015
3. Francesco Sapio, Getting Started with Unity 5.x 2D Game Development, 2017,
4. Tommaso Lintrami, Unity 2017 Game Development Essentials, Packt Publishing, 2017
5. Joseph Hocking. Unity in Action- Multiplatform Game Development in C#, Manning Publications, 2015
6. Penny de Byl, Holistic Game Development with Unity, CRC Press, 2017
7. Alan Thorn, Mastering Unity 5.x, Packt Publishing, 2017
8. Jonathan Linowes, Unity Virtual Reality Projects, , Packt Publishing, 2015

8.2 Seminar	**Teaching methods**	**Observations**
8.3 Laboratory		
GameMaker Studio Interface, sprites, editor	Laboratory work, Coding exercises	
GameMaker Studio – drag and drop, Pong Game	Laboratory work, Coding exercises	
GameMaker Studio – Game Maker Language	Laboratory work, Coding exercises	
GameMaker Studio Application with sound and score	Laboratory work, Coding exercises	
Unity 5.x Interface and entities	Laboratory work, Coding exercises	
Unity 5.x 2D Applications, sprites, C# scripting	Laboratory work, Coding exercises	
Unity 5.x 3D navigation, shooting	Laboratory work, Coding exercises	
Unity 5.x Assets	Laboratory work, Coding exercises	
Unity 5.x Character Animation	Laboratory work, Coding exercises	
Unity 5.x Environment, sound	Laboratory work, Coding exercises	
Unity 5.x Leap Motion Interface	Coding exercises	
Unity 5.x Virtual Reality	Coding exercises	
Unity 5.x Application	Project development, Help, Hints	Students are working for the project application
Unity 5.x Application	Project development, Help, Hints	Students are working for the project application

Bibliography

1. Mr Ben G Tyers, Practical GameMaker: Studio: Language Projects, Apress, 2016
2. Nathan Auckett, GameMaker Essentials, Packt Publishing, 2015
3. Francesco Sapio, Getting Started with Unity 5.x 2D Game Development, 2017,
4. Tommaso Lintrami, Unity 2017 Game Development Essentials, Packt Publishing, 2017
5. Joseph Hocking. Unity in Action- Multiplatform Game Development in C#, Manning Publications, 2015
6. Penny de Byl, Holistic Game Development with Unity, CRC Press, 2017
7. Alan Thorn, Mastering Unity 5.x, Packt Publishing, 2017
8. Jonathan Linowes, Unity Virtual Reality Projects, , Packt Publishing, 2015
9. https://www.yoyogames.com/learn
10. https://unity3d.com/learn

9. Subject's relevance to the epistemic community, professional associations and representative employers in fields significant for the program

- The subject introduces games and simulations development, together with problems related to human computer interfaces and real time processing. The games industry is young, but it is well develop in Romania, a leading provider of workforce in gaming industry, with employers like Ubisoft, Electronic Arts and many others. The humankind enters these years the world of full VR and experience in this field constitutes a big advantage not only on the labor market for game developers, but for software in general.

10. Assessment

Activity	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Weight in final grade
10.4 Course			
10.5 Seminar			
10.6 Laboratory/Project	Presence Course+Laboratory	Presence list	10%
	Homework 1 GameMaker	Homework assessment (Moodle)	25%
	Homework 2 3D Application in Unity 5.x	Homework assessment (Moodle)	25%
	Project – Game or Simulation	Defending the project. Printed sample	40%
10.7 Minimal standard of performance			
50% of points			

Programming Paradigms

1. Information about the program

1.1 Higher education institution	University POLITEHNICA of Bucharest
1.2 Faculty	Faculty of Engineering in Foreign Languages
1.3 Department	Department of Engineering in Foreign Languages
1.4 Field of study	Computers and Information Technology
1.5 Study cycle	Master
1.6 Program / Qualification	Software Engineering

2. Data about the subject

2.1 Name of subject			Programming Paradigms				
2.2 Course holder							
2.3 Seminar holder							
2.4 Laboratory/project holder							
2.5 Year of study	1	2.6 Semester	1	2.7 Evaluation type	V	2.8 Subject type	DO

3. Estimated time (hours per semester) of didactic activities

3.1 Number of hours per week	4	course hours	2	project	1	laboratory	1
3.2. Number of hours per semester	56	course hours	28	project	14	laboratory	14
3.3.Distribution of spend time:							**h.**
Study of textbooks, bibliography and course notes							14
Supplementary study in library, on electronic platforms, on the fieldwork							
Preparation of seminars/laboratories, home assignments, papers, portfolios, essays							10
Tutoring							2
Examinations							2
Other activities							

3.4 Total hours of individual study	**28**	
3.5 Total hours per semester	**84**	
3.6 Number of credits	**4**	

4. Preconditions (where relevant)

4.1 curriculum- related	• Programming Languages
4.2 competence - related	•

5. Facilities and equipment (where relevant)

5.1 for the course	• Projector, blackboard/whiteboard
5.2 for the course seminar	•
5.3 for the laboratory/project	• Laboratory with computers • Internet connection • Development boards with sensors and communication

	capabilities

6. Specific competences acquired

Professional competences	Sa cunoasca limbajele de programare pentru a fi efectiv si eficient in programare, inclusiv in crearea de codice, componente si servicii, si integrarea de numeroase subsisteme Know programming languages in order to be effective and efficient in programming, including in creading codices, components, services and in integrating numerous subsystems
Transversal competences	Organize software production processes using specific tools, components and services as well as abstract elements such as software algorithms and architectures.

7. Course objectives (as resulting from the grid of specific competences)

7.1 Subject general goal	• Understand the key elements and differences between programming languages based on syntax and semantics
7.2 Specific objectives	• Learn the imperative programming paradigm • Learn the object oriented programming paradigm • Learn the functional programming paradigm • Learn the declarative programming paradigm

8. Content

8.1 Course	**Teaching methods**	**Observations**
Introduction to programming paradigms	Blackboard, projector, Moodle	2
Abstract machines		2
Memory management		2
Data abstraction		2
Control abstraction		2
Imperative programming - Java		4
Object Oriented programming– Java		4
Functional programming– Scala		6
Declarative programming- Prolog		4

Bibliography

1. C. Horstmann, G. Cornell, “Core Java 2”
2. J. Bloch, C. Persuati, “Effective Java”
3. S. McConnell, “Code Complete”
4. M. Gabbrielli, S. Martini, “Programming Languages: Principles and Paradigms”

A.V. Aho “Compilers: Principles, Techniques and Tools”

8.2 Project	**Teaching methods**	**Observations**
Implement a complex software project that uses three programming paradigms (imperative, functional, declarative)	Moodle, individual work at computer	
8.3 Laboratory		

Java introduction		2 hours
Variables		2 hours
Methods		2 hours
Object Oriented Programming		**4** hours
Scala introduction		**2** hours
Recursivity in Scala		**4** hours
Generic types in Scala		**2** hours
Lists, sets in Scala		**2** hours
Pattern matching in Scala		**2** hours
Prolog introduction		**2** hours
Inferences in Prolog		**2** hours
Recursivity, lists in Prolog		**2** hours

Bibliography

1. C. Horstmann, G. Cornell, "Core Java 2"
2. J. Bloch, C. Persuati, "Effective Java"
3. S. McConnell, "Code Complete"
4. M. Gabbrielli, S. Martini, "Programming Languages: Principles and Paradigms"

A.V. Aho "Compilers: Principles, Techniques and Tools"

-

9. Subject's relevance to the epistemic community, professional associations and representative employers in fields significant for the program

- The course and laboratory were prepared after extensive study of similar programs offered by prestigious universities and adapted to be integrated in the current study program.

10. Assessment

Activity	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Weight in final grade
10.4 Course	Knowing the theory	Written exam	40
10.5 Project	Implement a software project that uses the three paradigms	Present in class	30
10.6 Laboratory/Project	Attendance + homework + activity	Oral examination	30
10.7 Minimal standard of performance			
• Achieve over 50% in score at the final exam • Present the project before the exam			

Formal Models in Software Engineering

1. Information about the program

1.1 Higher education institution	University POLITEHNICA of Bucharest
1.2 Faculty	Faculty of Engineering in Foreign Languages
1.3 Department	Department of Engineering in Foreign Languages
1.4 Field of study	Computers and Information Technology
1.5 Study cycle	Master
1.6 Program / Qualification	Software Engineering

2. Data about the subject

2.1 Name of subject			Formal Models in Software Engineering				
2.2 Course holder							
2.3 Seminar holder							
2.4 Laboratory/project holder							
2.5 Year of study	3	2.6 Semester	1	2.7 Evaluation type	C	2.8 Subject type	DPA/DO

3. Estimated time (hours per semester) of didactic activities

3.1 Number of hours per week	3	course hours	1	seminar	1	laboratory	1
3.2. Number of hours per semester	42	course hours	14	seminar	14	laboratory	14
3.3.Distribution of spend time:							**h.**
Study of textbooks, bibliography and course notes						20	
Supplementary study in library, on electronic platforms, on the fieldwork						10	
Preparation of seminars/laboratories, home assignments, papers, portfolios, essays						10	
Tutoring						3	
Examinations						3	
Other activities							

3.4 Total hours of individual study	**66**	
3.5 Total hours per semester	**108**	
3.6 Number of credits	**4**	

4. Preconditions (where relevant)

4.1 curriculum- related	• Introduction to Information Technology • Data Structures and Algorithms
4.2 competence - related	•

5. Facilities and equipment (where relevant)

5.1 for the course	• Overhead Projector
5.2 for the course seminar	•
5.3 for the laboratory/project	• 20 PC

6. Specific competences acquired

Professional competences	• Apply design and development methods and techniques as appropriate to realize solutions along the whole life-cycle of the software product
	• Understand and be able to use specific tools, components, and frameworks and also abstract elements such as algorithms and architectures • Organize and lead development teams, including team-building and negotiation • Serve as an agent of change for introducing new technology

7. Course objectives (as resulting from the grid of specific competences)

7.1 Subject general goal	• Scientific foundations for software engineering depend on the use of precise, abstract models for characterizing and reasoning about properties of software systems. This course considers many of the standard models for representing sequential and concurrent systems, such as grammars, automata, state machines, formal models in Spin and Promela. It shows how different logics can be used to specify properties of software systems, such as functional correctness, deadlock freedom, and internal consistency. Concepts such as composition mechanisms, abstraction relations, invariants, non-determinism, inductive definitions, operational and denotational descriptions are recurrent themes throughout the course.
7.2 Specific objectives	• How to use formal specification methods in software
	development • Formal reasoning on program correctness • Modeling and formal specification of software activities .

8. Content

Course	Teaching methods	Observations
Foundations 1.1. What's a formal model? 1.2. Logic and Proof Techniques 1.3. Sets, Relations, Maps, Functions 1.4. Graphs 1.5. Abstract Data Types	Lecturing	
Languages 2.1. Formal Systems 2.2. Grammars and Languages 2.3. Semantics Specifications 2.4. Automata and Languages	Lecturing	
State Machine 3.1. Basic Concepts 3.2. Variations of State Machines	Lecturing	
Models of Computing Systems 4.1. Introduction to Spin Promela 4.2. Formal Modelin in Promela 4.3. Verification and Validation in Spin/Promela 4.4. The Vienna Development Method 4.5. Operational vs. denotational semantics in VDM	Lecturing	

Bibliography

- Peter Linz, An Introduction to Formal Languages and Automata, 2006
- J. Glenn Brookshear, Theory of Computation: Formal Languages, Automata, and ComplexityJ, 1989
- John E. Hopcroft and Rajeev Motwani, Introduction to Automata Theory, Languages, and Computation, 2006
- Nagpal, Formal Languages and Automata Theory, 2012

8.2 Seminar	Teaching Method	Observation
Deterministic and Non-Deterministic Finite Automata	Seminar work	
Regular expression algorithms	Seminar work	
Grammars and Derivations	Seminar work	
8.3 Laboratory		
Introduction to Promela	Laboratory Work	
Introduction to Spin feature	Laboratory Work	

Elaboration of a complex Formal Model in Promela	Laboratory Work	
Simulation and Verification in Spin	Laboratory Work	

Bibliography

- Elaine A. Rich, Automata, Computability and Complexity: Theory and ApplicationsSep 28, 2007
- John E. Hopcroft and Rajeev Motwani, Introduction to Automata Theory, Languages, and Computation, 2006
- "Programare Functionala, O perspectiva pragmatica", C. Giumale, Editura tehnica, 1997
- "Z An Itroduction to Formal Methods", A. Diller, John Wiley & Sons, 1994
- "Systematic Software Development using VDM", C. B. Jones, Prentice Hall 1990
- "Concurrency: State Models and Java Programs", by Magee and Kramer [MK99]. .
- "Concepts and Notations for Concurrent Programming," Andrews and Schneider. Computing Surveys, Vol. 15, No. 1, March 1983.
- "Formal Methods: State of the Art and Future Directions", ACM Computing Surveys, Vol. 28, No. 4, December 1996, pp. 626-643. Available as CMU-CS-96-178.
- "Statecharts: a visual formalism for complex systems." D. Harel. Science of Computer Programming, 8:231-274, 1987.
- High-level Petri Nets: Theory and Application. K. Jensen and G. Rozenberg (eds.) Springer-Verlag, 1991.
- Concurrency: State Models and Java Programs, J. Magee and J. Kramer. Wiley, 1999.
- "Petri Nets." J. L. Peterson. ACM Computing Surveys, Sept 1977.
- The Z Notation: A Reference Manual, J. M. Spivey. Prentice-Hall International, 1989.
- Using Z: Specification, Refinement, and Proof, J. Woodcock and J. Davies. Prentice Hall 1996. Available from http://www.usingz.com/

9. Subject's relevance to the epistemic community, professional associations and representative employers in fields significant for the program

- Formal Modeling is a building block for checking the correctness of a software system – especially its architecture. Formal modeling showed its usefulness in protocol algorithms used in a distributed environment.

10. Assessment

Activity	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Weight in final grade
10.4 Course	Course Presence and Activities	Presence and Activities Evaluation	10%
	Final Examinations	Written Exam	40%
10.5 Seminar			
	Seminary Assignments	Assignments Correction	25%
10.6	Laboratory Assignment	Assignments Correction	10%
Laboratory/Project	Project	Project Evaluation	15%
10.7 Minimal standard of performance			
• **Minimum 50% of the marks from Seminaries Assignments and Course Activities part (3 points out of 6)** • **Minimum of 50 % from final examination (2 points out of 4)**			

Technologies for Big Data Analysis

1. Information about the program

1.1 Higher education institution	University POLITEHNICA of Bucharest
1.2 Faculty	Faculty of Engineering in Foreign Languages
1.3 Department	Department of Engineering in Foreign Languages
1.4 Field of study	Computers and Information Technology
1.5 Study cycle	Master
1.6 Program / Qualification	Software Engineering

2. Data about the subject

2.1 Name of subject			Technologies for Big Data Analysis				
2.2 Course holder							
2.3 Seminar holder							
2.4 Laboratory/project holder							
2.5 Year of study	2014	2.6 Semester	I	2.7 Evaluation type	E	2.8 Subject type	DAP/DO

3. Estimated time (hours per semester) of didactic activities

3.1 Number of hours per week	3	course hours	2	seminar		laboratory	1
3.2. Number of hours per semester	52	course hours	28	seminar		laboratory	14
3.3.Distribution of spend time:							**h.**
Study of textbooks, bibliography and course notes							24
Supplementary study in library, on electronic platforms, on the fieldwork							14
Preparation of seminars/laboratories, home assignments, papers, portfolios, essays							14
Tutoring							4
Examinations							10
Other activities							

3.4 Total hours of individual study	**66**	
3.5 Total hours per semester	**108**	
3.6 Number of credits	**4**	

4. Preconditions (where relevant)

4.1 curriculum- related	• Java programming, C++, Python, SQL
4.2 competence - related	• Architecture of Enterprise Information Systems, Business Intelligence

5. Facilities and equipment (where relevant)

5.1 for the course	• Overhead projector, internet connection
5.2 for the course seminar	• Workstations min 8 GB RAM
5.3 for the laboratory/project	• LAN between workstation to make a distribute cluster

6. Specific competences acquired

Professional competences	• Programming languages awareness for effective programming, including code, components and services creation, and integration of multiple subsystems
Transversal competences	• Understand and be able to use specific tools, components, and frameworks and also abstract elements such as algorithms and architectures • Organize and lead development teams, including team-building and negotiation • Serve as an agent of change for introducing new technology

7. Course objectives (as resulting from the grid of specific competences)

7.1 Subject general goal	• Familiarize the student with a framework for storing, processing and analyzing "Big Data"
7.2 Specific objectives	• Learn concepts about Hadoop Distributed File System and related concepts, applications and languages, including NoSQL

8. Content

8.1 Course	Teaching methods	Observations
What is Big Data – Use Cases and Intro to Apache Hadoop	Instructor Led Training	In class, Live Virtual Class
The Hadoop Ecosystem		
Writing a MapReduce Program in Java		
Delving Deeper into the Hadoop API		Combiner, Distributed cache
Practical Development Tips and Techniques		Strategies for debugging MapReduce code
Partitioners and reducers		Writing custom Partitioners
Common MapReduce Algorithms		How to sort, search, index and compute TF-IDF
Hadoop Tools for Data Acquisition		Use Sqoop and Flume
Introduction of NoSQL & Spark		Overview of NoSQL concepts and the
		Apache Spark framework
Features for data acquisition, storage and analysis		Usage of Pig and Hive
Multi Dataset operations with Pig		Usage of grouping, combining, join, concatenating and splitting
Introduction to Hive		How does Hive differ from RDBMS?
Text processing with Hive		Emphasize the need to analyze unstructured and semi-structured data
Introduction to Impala		How to achieve a trade off between high speed and cost for interactive/ad hoc queries in data analysis

Bibliography 1. Putting Big Data to Work: Opportunities for Enterprises, by Brett Sheppard, *© 2011 GigaOM* pro.gigaom.com *2.* The Big Book of Big Data, A field guide for Industry-based Big Data Opportunities, *© 2013 Oracle Inc., 1-st Edition* 3. Hadoop: The Definitive Guide, by Tom White, 3-rg Edition, O'Reilley 2012, ISBN-13: 978-1449311520 4. HBase, The Definitive Guide, by Larss George, 1-st Edition, O'Reilley 2011, ISBN-13: 978-1449396107 5. Programming Pig, by Alan Gates, 1-st Edition, O'Reilley 2011, ISBN-13: 978-1449302641 6. Programming Hive, by E. Capriolo, D. Wampler, J. Rutherglen, 1-st Edition, O'Reilley 2012, ISBN-13: 978-1449319335 7. Oracle NoSQL Database: Real-time Big Data Management for the Enterprise, by M. Alam, A. Muley, C. Kadaru, A. Joshi, 1-st Edition, Oracle Press 2013, ISBN-13: 978-0071816533 8. Oracle Big Data Handbook, by T. Plumkett & B. Macdonald, 1-st Edition, Oracle Press 2013, ISBN-13: 978-0071827263		
8.2 Seminar	**Teaching methods**	**Observations**
8.3 Laboratory		
Using HDFS		
Running a MapReduce Job		
Writing a MapReduce Java and Streaming Programs and Testing with MRUnit framework		
Using ToolRunner and Passing Parameters; using a Combiner		
Testing with LoalJobRunner, Logging and using Counters and a Map-Only Job		
Writing a Partitioner, implementing a Customr WritableComparable and using SequenceFiles and File Compression		
Creating an Inverted Index and calculating Word Co-Occurance		
Running an Oozie Workflow and exploring a Secondary Sort Example		
Using Pig for ETL Processing and Analyzing Ad Campaign Data with Pig		
Analyze Disparate Data Sets with Pig and extend Pig with Streaming and UDFs		
Running Hive Queries from the Shell, Scripts and Hue and performing Data Management with Hive		
Gaining Insight with Sentiment Analysis and Transforming Data with Hive		
Interactive Analysis with Impala		
Bibliography www.claudera.com		

9. Subject's relevance to the epistemic community, professional associations and representative employers in fields significant for the program

- BigData is the emerging domain dealing with the global digital disruption, at the confluence of Language programming in distributed systems, Business Intelligence/Advanced Analytics, Data Modeling and Statistics. With use cases in most activity fields, Big Data is the new paradigm in Information Systems
- Big Data is one of the 4 pillars for Horizon 2020 Research program of EU
- Mc Kinsey mentions a shortage of 140000-190000 people with deep analytical skills in US only
- UEFISCDI has established in 2013 that Big Data is one of the strategic directions for research in Romania

10. Assessment

Activity	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Weight in final grade
10.4 Course	Knowledge of the courseware content	Quiz of 30 questions randomly choosen from a datanase of 150 questions	70%
10.5 Seminar			
10.6 Laboratory/Project	Completion of two home assignments	Achievement of the two home assignments is a precondition to enter the final evaluation test	30%
10.7 Minimal standard of performance			
• 50% correct answer to the final Quiz			

Model Driven Software Engineering

1. Information about the program

1.1 Higher education institution	University POLITEHNICA of Bucharest
1.2 Faculty	Faculty of Engineering in Foreign Languages
1.3 Department	Department of Engineering in Foreign Languages
1.4 Field of study	Computers and Information Technology
1.5 Study cycle	Master
1.6 Program / Qualification	Software Engineering

2. Data about the subject

2.1 Name of subject			Model Driven Software Engineering				
2.2 Course holder							
2.3 Seminar holder							
2.4 Laboratory/project holder							
2.5 Year of study	1	2.6 Semester	2	2.7 Evaluation type	E	2.8 Subject type	DAP/DO

3. Estimated time (hours per semester) of didactic activities

3.1 Number of hours per week	3	course hours	2	seminar	-	laboratory	1
3.2. Number of hours per semester	42	course hours	28	seminar		laboratory	14
3.3.Distribution of spend time:							**h.**
Study of textbooks, bibliography and course notes						12	
Supplementary study in library, on electronic platforms, on the fieldwork						8	
Preparation of seminars/laboratories, home assignments, papers, portfolios, essays						14	
Tutoring						4	
Examinations						4	
Other activities							

3.4 Total hours of individual study	**42**	
3.5 Total hours per semester	**84**	
3.6 Number of credits	**4**	

4. Preconditions (where relevant)

4.1 curriculum- related	•
4.2 competence - related	• Object oriented modeling

5. Facilities and equipment (where relevant)

5.1 for the course	• Room with video projector
5.2 for the course seminar	•
5.3 for the laboratory/project	• Computer laboratory

6. Specific competences acquired

Professional competences	• Apply design and development methods and techniques as appropriate to realize solutions along the whole life-cycle of the software product
Transversal competences	• Understand and be able to use specific tools, components, and frameworks and also abstract elements such as algorithms and architectures • Organize and lead development teams, including team-building and negotiation • Serve as an agent of change for introducing new technology

7. Course objectives (as resulting from the grid of specific competences)

7.1 Subject general goal	• The general goal is to show that models have overtaken their declarative role in software development and have become imperative in certain contexts, being used for driving applications, similarly as programs, but at a higher level of abstraction.
7.2 Specific objectives	• Present metamodels as languages for defining models • Describe software architectures specific for rendering the models executable • Show the possibilities to transform and compose models and metamodels. • Prove that software development may be leveraged by using various models, metamodels, languages, by getting familiar with the definition and interpretation of models • Define metamodels and generating editors corresponding to them • Familiarize the student with some software development environments that support MDE.

8. Content

8.1 Course	Teaching methods	Observations
Principles of Model Driven Engineering	- Presentation on slides - Discussions - Formative tests	
Model Driven Architecture and standards		
UML Metamodel and the possibility to extend it with profiles		
Metamodeling Languages: MOF, Ecore, XMF, GME language		
Domain Specific Languages		
Software Environments for Modeling and Metamodeling		
Using DSLs in Product Line Architecture.		

Generative Programming.		
Using workflow models in Service Oriented Architectures		
Separation of concerns. Domain Models		
Types of model transformations		
Model markers and annotations		
Automatic code generation		
Composing models and metamodels		

Bibliography

St.J. Mellor, K. Scott, A. Uhl, D. Weise, MDA Distilled: Priciples of Model-Driven Architecture, Addison Weesley, 2004

J. Greenfield, K. Short, Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools, Wiley Publishing, Inc., 2004

J. Estublier, A. D. Ionita, G. Vega, Relationships for Domain Reuse and Composition, Journal of Research and Practice in Information Technology, 38, 4, ISSN 1443-458X, pp. 287-301, 2006

T. Clerk, P. Sammut, J. Willams, Applied Metamodelling. A Foundation for Language Driven Development, Ceteva, 2008

D. Gaševic, D. Djuric, V. Devedžic, Model Driven Engineering and Ontology Development, Springer, 2nd ed., 2009

Ian Sommerville, Software Engineering, Editia a 9-a, Addison-Wesley, 2010

J. Estublier, A.D. Ionita, T. Nguyen, Code Generation for a Bi-dimensional Composition Mechanism, In "Software engineering techniques : Third IFIP TC 2 Central and East European Conference, CEE-SET 2008, Brno, Czech Republic, October 13-15, 2008, revised selected papers", Z. Huzar et al. (Eds.), Lecture Notes in Computer Science, LNCS 4980, pp. 171-185, Springer, 2011

A.D. Ionita, J. Estublier, "Business Process Modeling and Automation with General and Domain Specific Languages", In Business Process Modeling: Software Engineering, Analysis and Applications, Jason A. Beckmann Ed., Seria Business Issues, Competition and Entrepreneurship, Nova Science Publishers, 2011

A.D. Ionita, A. Olteanu, T. Ionescu, L. Dobrica, Automatic Transformations for Integrating Instrument Models across Technological Spaces, Romanian Journal of Information Science and Technology, Volume 14, Number 1, ISSN: 1453-8245, The Publishing House of the Romanian Academy, 2011, pp. 51-66

OMG, OMG Unified Modeling Language TM (OMG UML), Version 2.5, September 2013, disponibil la www.omg.org

A.D. Ionita, M. Litoiu, G. Lewis (Editors) Migrating Legacy Applications: Challenges in Service-Oriented Architecture and Cloud Computing Environments, IGI Global, 2013

8.2 Seminar	**Teaching methods**	**Observations**
8.3 Laboratory		
Study examples of models and metamodels, in order to understand the necessity of metamodeling	- Explanations at the beginning of the laboratory - Assisted individual work - Study of examples - Use of a predefined structure for the specifications - Homework verification and evaluation	
Define models starting from a given metamodel		
Define a metamodel (DSL) for a given application domain		
Use a metamodeling environment for defining a metamodel and generating an editor of models conforming to it		
Configure the metamodel interpreter		

Bibliography

GME Manual and User Guide, 2000-2014 Vanderbilt University

9. Subject's relevance to the epistemic community, professional associations and representative employers in fields significant for the program

- The content is aligned to the specifications adopted by an international industrial standards consortium (OMG) and to the competencies required by software companies. Apart from general modeling languages, the course also approaches domain specific languages, which are used in many industrial settings based on product lines.

10. Assessment

Activity	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Weight in final grade
10.4 Course	Specific criteria for each question of the examination	Final verification (exam)	40
10.5 Seminar			
10.6 Laboratory/Project	All the indicated topics approached Coherent concepts and relationships	Evaluation of a case study	10
	Executable modeling paradigm Correspondence between the case study and the model conforming to the modeling paradigm Specific concrete syntax defined	Verification of individual work and evaluation of the customized modeling paradigm	30
	Marks for participation to the class activities	Average of marks	10
	Evaluation of the configuration for a given modeling paradigm	Laboratory test	10
10.7 Minimal standard of performance			
• Accumulation of minimum 50 points			

Distributed Software Engineering

1. Information about the program

1.1 Higher education institution	University POLITEHNICA of Bucharest
1.2 Faculty	Faculty of Engineering in Foreign Languages
1.3 Department	Department of Engineering in Foreign Languages
1.4 Field of study	Computers and Information Technology
1.5 Study cycle	Master
1.6 Program / Qualification	Software Engineering

2. Data about the subject

2.1 Name of subject			Distributed Software Engineering				
2.2 Course holder							
2.3 Seminar holder							
2.4 Laboratory/project holder							
2.5 Year of study	1	2.6 Semester	2	2.7 Evaluation type	E	2.8 Subject type	DAP/DA

3. Estimated time (hours per semester) of didactic activities

3.1 Number of hours per week	4	course hours	2	seminar		project	2
3.2. Number of hours per semester	56	course hours		seminar		project	28
3.3.Distribution of spend time:						**52**	**h.**
Study of textbooks, bibliography and course notes						12	
Supplementary study in library, on electronic platforms, on the fieldwork						12	
Preparation of seminars/laboratories, home assignments, papers, portfolios, essays						25	
Tutoring							
Examinations						3	
Other activities							

3.4 Total hours of individual study	**52**	
3.5 Total hours per semester [1]	**108**	
3.6 Number of credits	**4**	

4. Preconditions (where relevant)

4.1 curriculum- related	• Attending and/or passing the following courses: Parallel and distributed algorithms, Computer networks, Distributed systems
4.2 competence - related	• Operate with scientific, engineering, and information technology concepts, Solve problems by using instruments of the computer science and engineering field

Facilities and equipment (where relevant)

5.1 for the course	•
5.2 for the course seminar	•
5.3 for the laboratory/project	•

5. Specific competences acquired

Professional competences	• Apply design and development methods and techniques as appropriate to realize solutions along the whole life-cycle of the software product • Programming languages awareness for effective programming, including code, components and services creation, and integration of multiple subsystems
Transversal competences	• Understand and be able to use specific tools, components, and frameworks and also abstract elements such as algorithms and architectures • Organize and lead development teams, including team-building and negotiation • Serve as an agent of change for introducing new technology

6. Course objectives (as resulting from the grid of specific competences)

7.1 Subject general goal	• Learning and integrating the main concepts, principles, models, and techniques related to distributed program systems development. Capability to use this knowledge in modeling, design of software components for distributed systems. Implementing middleware programs using current technologies. Evaluation how the developed systems satisfy the specification criteria and performance optimization by using specific instruments and engineering methods.
7.2 Specific objectives	• Study actual problems in the domain of distributed processing over computer networks. • Research new solutions to solve complex problems in distributed systems, related to assuring interprocess communication, data replication for increasing performance, ensuring consistency, fault tolerance, and security in Web based systems. • Study heterogeneous systems based on objects, mobile networks and mobile agents. • Acquire practical abilities needed for the design, implementation, and evaluation of distributed systems components.
	• Identification of concrete problems related to actual distributed systems and finding high performance solutions. • Effective use of design and implementation instruments for distributed systems. • Deployment, exploitation, and maintenance of distributed system programs.

7. Content

8.1 Course	Teaching methods	Observations

Introduction. Models and architectures of dynamic large scale distributed systems (client-server, service oriented, peer-to-peer and others). Specific design requirements: scalability, transparency, and performance.	Face to face teaching	The course provides teaching, supplemented by discussing recommended paper in class. Course teaching is done with PPT presentations. For paper discussions, papers are made accessible on the course site. Students will shortly present personal opinions about the papers under discussion. On course site are provided: - teaching materials (presentation slides) - fragments from books and papers authored by the teacher - papers to be discussed in the class - information about course administration - information about the points acquired by students during the semester - discussion forum.
P2P Systems. Internet Architectures and technologies for content distribution (client/server, multicast, P2P). Search services, hash tables. Search and download performance in P2P Systems. Gossip protocols. P2P streaming. Dynamic scalable efficient content replication techniques. Anonymity. Reputation management in P2P systems. Self-* properties in P2P systems.		
Event driven distributes systems. Architectures, components. Complex event processing, detecting events templates. ECA (event, condition, action) and finite state machines with interval timestamps. Intelligent engines for event processing. Events and increasing reactivity of RIA (Rich Internet Applications). Event based Collaborative Applications.		
Cloud computing. Resource provisioning. Optimizing resource use. Energy management. Traffic management. Data Security. Software frameworks for performance optimization. Storage Technologies and data management. Data and computation intensive models: MapReduce.		
Context based distributed systems. Context information sensing, transmission, and processing. Context models; ontology-based models. Context based autonomous systems. Resource discovery. Historical context data. Security and privacy.		

Bibliography

1. A.S. Tanenbaum, M. van Steen. Distributed Systems. Principles and paradigms, Prentice Hall 2007
2. George Coulouris, Jean Dollimore,Tim Kindberg, Gordon Blair. Distributed Systems Concepts and Design (Fifth Edition), Addison-Wesley, 2012
3. L. Shklar, R. Rosen. Web Application Architecture, John Wiley & Sons, 2003
4. E. Cerami. Web Services, O'Relly 2002
5. T.B. Passin. Explorer's Guide to the Semantic Web, Manning 2004
6. Valentin Cristea, Ciprian Dobre, Corina Stratan, Florin Pop, Alexandru Costan, "Large-scale Distributed Computing and Applications: Models and Trends", Ed. Information Science Publishing, ISBN : 978-1615207039, 390 pg., April 2010.
7. Cristea, Valentin and Dobre, Ciprian and Pop, Florin. Context-Aware Environments for the Internet of Things. Chapter in Internet of Things and Inter-cooperative Computational Technologies for Collective Intelligence (Nik Bessis, Fatos Xhafa,

Dora Varvarigou, Richard Hill, Maozhen Li). Springer Berlin Heidelberg, vol. 460, pp. 25--49, August 2013 8. Set of up-to-date papers offered to students on the course site.		
8.2 Seminar	**Teaching methods**	**Observations**
Requirements analysis	Individual or group (2-3 students) projects	On the course site the following information is available: - support documentation for projects development - information regarding seminar and project administration - discussion forum.
Documentary research of state-of-the art in the domain		
Design model elaboration		
Development instruments selection		
Implementation		
Performance analysis		
Conclusions, further work, results exploitation		
8.3 Project		
Idem Seminar		Projects are posted on the site course for evaluation and ranking.
Bibliography Idem Course		

8. Subject's relevance to the epistemic community, professional associations and representative employers in fields significant for the program

-

9. Assessment

Activity	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Weight in final grade
10.4 Course	Responses' correctness	Topics discussions in the class	40%
	Correctness of the responses to exam queries	Written exam	10%
10.5 Seminar			
10.6 Project	Correctness and completeness of the responses	Tests during the semester	30%
	Correctness and completeness of the solutions	Project defense	20%
10.7 Minimal standard of performance			
• obtain 50 % of the points allocated to the Course (3.5 points out of 7) • obtain 50 % of the points allocated to project elaboration and final defense (1.5 points out of 3)			

Software Methodologies

1. Information about the program

1.1 Higher education institution	University POLITEHNICA of Bucharest
1.2 Faculty	Faculty of Engineering in Foreign Languages
1.3 Department	Department of Engineering in Foreign Languages
1.4 Field of study	Computers and Information Technology
1.5 Study cycle	Master
1.6 Program / Qualification	Software Engineering

2. Data about the subject

2.1 Name of subject			**Software Methodologies**				
2.2 Course holder							
2.3 Seminar holder			-				
2.4 Laboratory/project holder							
2.5 Year of study	1	2.6 Semester	2	2.7 Evaluation type	E	2.8 Subject type	DSI/DO

3. Estimated time (hours per semester) of didactic activities

3.1 Number of hours per week	3	course hours	2	seminar		laboratory	1
3.2. Number of hours per semester	42	course hours	28	seminar		laboratory	14
3.3.Distribution of spend time:							**h.**
Study of textbooks, bibliography and course notes							32
Supplementary study in library, on electronic platforms, on the fieldwork							
Preparation of seminars/laboratories, home assignments, papers, portfolios, essays							32
Tutoring							
Examinations							2
Other activities							

3.4 Total hours of individual study	**64**	
3.5 Total hours per semester [1]	**108**	
3.6 Number of credits	**4**	

4. Preconditions (where relevant)

4.1 curriculum- related	• Graduating one or more programming courses
4.2 competence - related	• Minimal background of Computer science

Facilities and equipment (where relevant)

5.1 for the course	•
5.2 for the course seminar	•
5.3 for the laboratory/project	•

5. Specific competences acquired

Professional competences	• Design appropriate software solutions, using responsible engineering approaches and applying theories and models that provide a basis for software design • Apply design and development methods and techniques as appropriate to realize solutions along the whole life-cycle of the software product
Transversal competences	• Understand and be able to use specific tools, components, and frameworks and also abstract elements such as algorithms and architectures • Organize and lead development teams, including team-building and negotiation • Serve as an agent of change for introducing new technology

6. Course objectives (as resulting from the grid of specific competences)

7.1 Subject general goal	• The course provides students with both a broad understanding of the space of current methodologies, and specific skills in using these methodologies. It provides methods, techniques and tools for systematic development of complex systems and software systems in particular
7.2 Specific objectives	• The course provides an overview of the evolution of the methodologies of systems/software development, including the latest development methodologies based on software components and service orientation.

7. Content

8.1 Course	**Teaching methods**	**Observations**
Introduction to Systems Engineering	slides	
Enterprise-Wide Information System Methodologies	slides	
Introduction to Software Engineering	slides	
Structured Information System Methodology	slides	
Object-Oriented Software Methodology	slides	

Bibliography

Derek Hatley, Peter Hruschka, Imtiaz Pirbhai, “Process for System Architecture and Requirements Engineering”, Dorset House Publ., 2000.

Ian Sommerville, Software Engineering. 8th Edition. Addison-Wesley 2007.

Ed Yourdon, Modern Structured Analysis, Prentice Hall, 1989.

Luca Dan Serbanati, “Integrating Tools for Software Development”, Prentice Hall, 1993.

Luca Dan Serbanati, Software Methodologies, Lecture Notes, 2014, http://www.serbanati.com/poli/index_smeth.php.

Thomas Erl, “Service-Oriented Architecture (SOA): Concepts, Technology, and Design”, Prentice Hall, 2005.

James Rumbaugh, Ivar Jacobson, Grady Booch, “The unified modeling language reference manual”, v. 1-3, Addison-Wesley, 1999-200,

8.2 Seminar	Teaching methods	Observations
8.3 Laboratory		
Introduction to Systems Engineering	Debate of the homework, exercise solving, laboratory work	
Enterprise-Wide Information System Methodologies	Debate of the homework, exercise solving, laboratory work	
Introduction to Software Engineering	Debate of the homework, exercise solving, laboratory work	
Structured Information System Methodology	Debate of the homework, exercise solving, laboratory work	
Object-Oriented Software Methodology	Debate of the homework, exercise solving, laboratory work	

Bibliography

Derek Hatley, Peter Hruschka, Imtiaz Pirbhai, “Process for System Architecture and Requirements Engineering”, Dorset House Publ., 2000.

Ed Yourdon, Modern Structured Analysis, Prentice Hall, 1989.

Rod Johnson, “Expert One-on-One J2EE Design and Development”, Wrox, 2002.

8. Subject’s relevance to the epistemic community, professional associations and representative employers in fields significant for the program

- Practical development of software requires an understanding of successful methods for bridging the gap between a problem to be solved and a working software system. This course focuses specifically on methods that guide the software engineer from requirements to code. For this it presents the most known software methodologies, that is those knowledge realms created around a development paradigm and formed from methods, techniques, rules, postulates, and tools to be used for software fabrication. They guide the software engineer in software development process from requirements identification to code generation and validation.

9. Assessment

Activity	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Weight in final grade
10.4 Course	Written examination	Questions of theoretical knowledge	30%
		Solving practical exercises	30%
10.5 Seminar			
10.6		Homework	20%

Laboratory/Project		Mini-project	20%
10.7 Minimal standard of performance			
• Exam admission criterium : only with a higher grade of 4 • Exam promotion with the final grade at least 5			

Computing in the Semantic Web

1. Information about the program

1.1 Higher education institution	University POLITEHNICA of Bucharest
1.2 Faculty	Faculty of Engineering in Foreign Languages
1.3 Department	Department of Engineering in Foreign Languages
1.4 Field of study	Computers and Information Technology
1.5 Study cycle	Master
1.6 Program / Qualification	Software Engineering

2. Data about the subject

2.1 Name of subject			Computing in the Semantic Web				
2.2 Course holder							
2.3 Seminar holder							
2.4 Laboratory/project holder							
2.5 Year of study	1	2.6 Semester	2	2.7 Evaluation type	E	2.8 Subject type	DPA/DO

3. Estimated time (hours per semester) of didactic activities

3.1 Number of hours per week	3	course hours	2	seminar	0	laboratory	1
3.2. Number of hours per semester	42	course hours	28	seminar	0	laboratory	14
3.3.Distribution of spend time:							**h.**
Study of textbooks, bibliography and course notes						10	
Supplementary study in library, on electronic platforms, on the fieldwork						10	
Preparation of seminars/laboratories, home assignments, papers, portfolios, essays						7	
Tutoring						10	
Examinations						3	
Other activities							

3.4 Total hours of individual study	**40**	
3.5 Total hours per semester [1]	**82**	
3.6 Number of credits	**4**	

4. Preconditions (where relevant)

4.1 curriculum- related	• Database, HTML, algorithm design, web services
4.2 competence - related	•

5. Facilities and equipment (where relevant)

5.1 for the course	•
5.2 for the course seminar	•
5.3 for the laboratory/project	•

6. Specific competences acquired

Professional competences	• Apply design and development methods and techniques as appropriate to realize solutions along the whole life-cycle of the software product • Programming languages awareness for effective programming, including code, components and services creation, and integration of multiple subsystems
	• Understand and be able to use specific tools, components, and frameworks and also abstract elements such as algorithms and architectures • Organize and lead development teams, including team-building and negotiation • Serve as an agent of change for introducing new technology

7. Course objectives (as resulting from the grid of specific competences)

7.1 Subject general goal	Designing and implementing semantic web applications aligned with the vision of "linked data" - perceived as the main evolution of the current social web.
7.2 Specific objectives	For course: - The semantic web is seen as a collection of accessible information that can be organized and used at a semantic level instead of using it at the syntactic and structural level. The course describes both a new generation of web standards and applications that have the ability to represent and use the semantics of specific information as well as the technologies needed to build such applications. The course will address the understanding of the Semantic Web from three perspectives: the theoretical aspects of information organization such as ontologies, taxonomies and semantic data modelling; aspects of understanding and creating information networks, and application-

	centered Web services semantics, semantic services for the business process, APIs and mashups. For applications: - The applications aim at building ontologies and writing the code that accesses them; representing XML data with appropriate semantic markings, which are obtained or derived from ontology; describing semantic relations between these elements using RDF and OWL; developing an application that discovers data and/or Web services based on semantic criteria.

8. Content

Course	Teaching methods	Observations
Introduction • Representation of knowledge • RDF • RDFS and OWL • The process of creating ontologies • Alignment of ontologies • SPARQL • Semantic web-specific tools • The social semantic web • Semantic Models (LSA - Latent Semantic Analysis and LDA - Latent Dirichlet Allocation, word2vec) • Semantic web services • Trends in the field	Face-to-face Lecturing	Course Material available in electronic format.

Bibliography

1. Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. Scientific american, 284(5), 28-37.
2. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., ... & Bizer, C. (2015). DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia. Semantic Web, 6(2), 167-195.
3. Maedche, A. (2012). Ontology learning for the semantic web (Vol. 665). Springer Science & Business Media.
4. Landauer, T. K., & Dumais, S. T. (1997). A solution to Plato's problem: the Latent Semantic Analysis theory of acquisition, induction and representation of knowledge. Psychological Review, 104(2), 211–240.
5. Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. Journal of Machine Learning Research, 3(4-5), 993–1022.
6. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representation in Vector Space. In Workshop at ICLR. Scottsdale, AZ.
7. M. C. Daconta, et al. The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management. John Wiley & Sons, Inc. 2003
8. D. Allemang, J. Hendler. Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL, Morgan Kaufmann, 2008
9. G. Antoniou, F. van Harmelen. A Semantic Web Primer, The MIT Press, 2nd Edition (Cooperative

Information Systems) 2008
10. L. Yu. Introduction to the Semantic Web and Semantic Web Services, Chapman & Hall, 2007
11. *** Course notes and slides

8.2 Seminar	**Teaching Method**	**Observation**
8.3 Laboratory		
Specifying an ontology	Laboratory Work	
Extracting data	Laboratory Work	
Implementing a semantic service	Laboratory Work	
Developing an application that uses semantic services	Laboratory Work	
Implement a given application	Laboratory Work	

Bibliography

The same as for the course

9. Subject's relevance to the epistemic community, professional associations and representative employers in fields significant for the program

Technologies for the semantic web are an important factor for processing information from the web, which today is very important in the success of many industrial, commercial enterprises. The course offers students the opportunity to get acquainted with such techniques and information processing from the web, a fact that is very important at the moment.

10. Assessment

Activity	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Weight in final grade
10.4 Course	The Correctness of solving the problems	Written Exam	40%
10.5 Seminar			
10.6 Laboratory/Project	Laboratory Assignment	Evaluating homework	60%
10.7 Minimal standard of performance			

- **Minimum 50% of the marks from Seminaries Assignments and Course Activities part (3 points out of 6)**
- **Minimum of 50 % from final examination (2 points out of 4)**

Advanced Topics in Computer Networks

1. Information about the program

1.1 Higher education institution	University POLITEHNICA of Bucharest
1.2 Faculty	Faculty of Engineering in Foreign Languages
1.3 Department	Department of Engineering in Foreign Languages
1.4 Field of study	Computers and Information Technology
1.5 Study cycle	Master
1.6 Program / Qualification	Software Engineering

2. Data about the subject

2.1 Name of subject			Advanced Topics in Computer Networks				
2.2 Course holder							
2.3 Seminar holder							
2.4 Laboratory/project holder							
2.5 Year of study	1	2.6 Semester	2	2.7 Evaluation type	E	2.8 Subject type	DPA/DO

3. Estimated time (hours per semester) of didactic activities

3.1 Number of hours per week	3	course hours	1	seminar		laboratory	2
3.2. Number of hours per semester	42	course hours	14	seminar		laboratory	28
3.3.Distribution of spend time:							**h.**
Study of textbooks, bibliography and course notes						20	
Supplementary study in library, on electronic platforms, on the fieldwork						20	
Preparation of seminars/laboratories, home assignments, papers, portfolios, essays						10	
Tutoring						3	
Examinations						3	
Other activities							

3.4 Total hours of individual study	**66**	
3.5 Total hours per semester [1]	**108**	
3.6 Number of credits	**4**	

4. Preconditions (where relevant)

4.1 curriculum- related	• Introduction to Information Technology • Data Structures and Algorithms
4.2 competence - related	• Programing Experience

5. Facilities and equipment (where relevant)

5.1 for the course	• Overhead Projector
5.2 for the course seminar	•
5.3 for the laboratory/project	• 20 PC

6. Specific competences acquired

Professional competences	• Apply design and development methods and techniques as appropriate to realize solutions along the whole life-cycle of the software product • Programming languages awareness for effective programming, including code, components and services creation, and integration of multiple subsystems
Transversal Competences	• Understand and be able to use specific tools, components, and frameworks and also abstract elements such as algorithms and architectures • Organize and lead development teams, including team-building and negotiation • Serve as an agent of change for introducing new technology

7. Course objectives (as resulting from the grid of specific competences)

7.1 Subject general goal	• The course provides students with advanced digital network concepts and principles. The course introduces students to internetworking, routing and network management. Students are provided with an opportunity to design and implement a network, network management and information routing throughout the network.
7.2 Specific objectives	• Ability to apply knowledge of Advanced Network Engineering including design, routing, management, security and performance and ability to understand and use industry standard tools used. • Ability to formulate and solve problems creatively, especially in network design, routing, management, security and performance.

8. Content

Course	Teaching methods	Observations
Review for networking basics and IP networks	Lecturing	
Introduction to wireless networks	Lecturing	
Introduction to algorithm design and optimization, and their applications in networking.	Lecturing	
Scheduling algorithms and MAC layer protocols (link layer)	Lecturing	
Routing algorithms and protocols (network layer)	Lecturing	
Congestion control algorithms and protocols (transport layer)	Lecturing	
Cross-layer design	Lecturing	
Quality of Service (QoS) provisioning	Lecturing	
Network security	Lecturing	

Bibliography

1. Computer Networks: A Systems Approach (4th Edition) by Larry Peterson and Bruce Davie. Morgan Kaufmann, 2007. ISBN: 0123705487.
2. Technical papers from major networking journals including IEEE/ACM Transactions on Networking, IEEE Transactions on Mobile Computing, IEEE Journals on Selected Areas in Communications, IEEE Transactions on Wireless Communications, ACM Transactions on Sensor Networks, Journal of Computer Networks and so on.
3. Technical papers from major networking conferences including IEEE INFOCOM, IEEE ICC, ACM MobiCom, ACM MobiHoc, ACM SenSys and so on.

8.2 Seminar	Teaching Method	Observation
8.3 Laboratory		
Network fundamentals	Laboratory Work	
wireless fundamentals	Laboratory Work	
CDMA, Bluetooth, sensor networks	Laboratory Work	
LAN, cellular networks	Laboratory Work	

Bibliography

Same as for the course

9. Subject's relevance to the epistemic community, professional associations and representative employers in fields significant for the program

- Computer Network is a building block for the building of a software system. Any modern computational system is mostly in a distributed form and it uses the network capabilities for the well-functioning of the

software. Therefore, skills acquired during this course represents an important building block for the making of software systems.

10. Assessment

Activity	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Weight in final grade
10.4 Course	Course Presence and Activities	Presence and Activities Evaluation	10%
	Final Examinations	Written Exam	40%
10.5 Seminar			
10.6 Laboratory/Project	Laboratory Assignment	Assignments Correction	50%
10.7 Minimal standard of performance			
• **Minimum 50% of the marks from Seminaries Assignments and Course Activities part (3 points out of 6)** • **Minimum of 50 % from final examination (2 points out of 4)**			

Software Architectures

1. Information about the program

1.1 Higher education institution	University POLITEHNICA of Bucharest
1.2 Faculty	Faculty of Engineering in Foreign Languages
1.3 Department	Department of Engineering in Foreign Languages
1.4 Field of study	Computers and Information Technology
1.5 Study cycle	Master
1.6 Program / Qualification	Software Engineering

2. Data about the subject

2.1 Name of subject			Software Architectures				
2.2 Course holder							
2.3 Seminar holder							
2.4 Laboratory/project holder							
2.5 Year of study	2	2.6 Semester	1	2.7 Evaluation type	E	2.8 Subject type	DAP/DO

3. Estimated time (hours per semester) of didactic activities

3.1 Number of hours per week	4	course hours	2	seminar		laboratory	2
3.2. Number of hours per semester	56	course hours	28	seminar		laboratory	28
3.3.Distribution of spend time:							**h.**
Study of textbooks, bibliography and course notes							24
Supplementary study in library, on electronic platforms, on the fieldwork							
Preparation of seminars/laboratories, home assignments, papers, portfolios, essays							24
Tutoring							
Examinations							4
Other activities							

3.4 Total hours of individual study	**52**	
3.5 Total hours per semester [1]	**108**	
3.6 Number of credits	**4**	

4. Preconditions (where relevant)

4.1 curriculum- related	• Graduating the Software Methodologies course
4.2 competence - related	• Background knowledge in systems engineering and computer programming

5. Facilities and equipment (where relevant)

5.1 for the course	•
5.2 for the course seminar	•
5.3 for the laboratory/project	•

6. Specific competences acquired

Professional competences	• Design appropriate software solutions, using responsible engineering approaches and applying theories and models that provide a basis for software design • Work effectively in interdisciplinary contexts, in particular to bridge the gap between computing technology and the clients business and to interpret and respect extra-technical constraints deriving from the business organization
Transversal competences	• Understand and be able to use specific tools, components, and frameworks and also abstract elements such as algorithms and architectures • Organize and lead development teams, including team-building and negotiation • Serve as an agent of change for introducing new technology

7. Course objectives (as resulting from the grid of specific competences)

7.1 Subject general goal	• The course teaches students knowledge and skills necessary for evaluating the architecture of existing systems and architectural design of new systems in accordance with current architectural paradigms.
7.2 Specific objectives	• The course aims to introduce: advanced software systems architectures, techniques for designing and implementing these architectures, formal models and notations for characterization and development of these architectures, tools for generating instances of architectures, and case studies of actual systems architectures.

8. Content

8.1 Course	Teaching methods	Observations
Software architectural styles	slides	
Component-based architectures	slides	
Middleware systems	slides	
Enterprise integration	slides	
Service-oriented architectures	slides	

Bibliography

1. D. Garlan, M. Shaw, "An Introduction to Software Architecture", Advances in Software Engineering and Knowledge Engineering", Volume I, World Scientific, 1993
2. P. Avgeriou, U. Zdun, “Architectural Patterns Revisited – A Pattern Language”, in: Proceedings of 10th European Conference on Pattern Languages of Programs (EuroPlop 2005), Irsee, Germany, July, 2005
3. Neil B. Harrison, Paris Avgeriou, Uwe Zdun, "Using Patterns to Capture Architectural Decisions," IEEE Software, vol. 24, no. 4, pp. 38-45, July/Aug. 2007.
4. http://www.oracle.com/technetwork/java/javaee/overview/index.html
5. http://www.omg.org/spec/CORBA/
6. http://www.enterpriseintegrationpatterns.com/
7. http://www.springframework.org/
8. Fred A. Cummins (2002). *Enterprise Integration: An Architecture for Enterprise Application and Systems Integration*. John Wiley & Sons. ISBN 0-471-40010-6
9. Gregor Hohpe - Bobby Woolf, "Enterprise Integration Patterns", The Addison-Wesley Professional, 2003
10. JSR 316: JavaTM Platform, Enterprise Edition 6 (Java EE 6) Specification http://jcp.org/en/jsr/detail?id=316
11. JSR 220: Enterprise JavaBeansTM 3.0 https://www.jcp.org/en/jsr/detail?id=220

8.2 Seminar	**Teaching methods**	**Observations**
8.3 Laboratory		
Software architectural styles	Debate of the homework, laboratory work	
Component-based architectures	Debate of the homework laboratory work	
Middleware systems	Debate of the homework, laboratory work	
Enterprise integration	Debate of the homework, laboratory work	
Service-oriented architectures	Debate of the homework, laboratory work	

Bibliography

1. http://www.oracle.com/technetwork/java/javaee/overview/index.html
2. http://www.omg.org/spec/CORBA/
3. http://www.enterpriseintegrationpatterns.com/
4. http://www.springframework.org/
5. JSR 316: JavaTM Platform, Enterprise Edition 6 (Java EE 6) Specification http://jcp.org/en/jsr/detail?id=316
6. JSR 220: Enterprise JavaBeansTM 3.0 https://www.jcp.org/en/jsr/detail?id=220

9. Subject’s relevance to the epistemic community, professional associations and representative employers in fields significant for the program

- The design of complex software systems requires skills in describing, evaluating and creating systems at architectural abstraction level. This course introduces architectural design of complex software systems in accordance with the latest trends in software technologies.

10. Assessment

Activity	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Weight in final grade
10.4 Course	Written examination	Questions of theoretical knowledge	25%
		Solving practical exercises	25%
10.5 Seminar			
10.6 Laboratory/Project		Homework	20%
		Project	30%
10.7 Minimal standard of performance			
•			

Software Project Management

1. Information about the program

1.1 Higher education institution	University POLITEHNICA of Bucharest
1.2 Faculty	Faculty of Engineering in Foreign Languages
1.3 Department	Department of Engineering in Foreign Languages
1.4 Field of study	Computers and Information Technology
1.5 Study cycle	Master
1.6 Program / Qualification	Software Engineering

2. Data about the subject

2.1 Name of Subject	**Software Project Management**						
2.2 Course holder							
2.3 Laboratory holder							
2.4 Year of study	6	2.5 Semester	1	2.6 Evaluation type	E	2.7 Subject type	DAP/DO

3. Estimated time (hours per semester) of didactic activities

3.1 Number of hours per week	4	course	2	seminar		laboratory	2
3.4 Number of hours per semester	56	course	28	seminar		laboratory	28
3.3.Distribution of spend time:							
Study of textbooks, bibliography and course notes							12
Supplementary study in library, on electronic platforms, on the fieldwork							10
Preparation of seminars/laboratories, home assignments, papers, portfolios, essays							10
Tutoring							7
Examinations							3
Other activities							

3.4 Total hours of individual study	52
3.5 Total hours per semester	108
3.6 Number of credits	5

4. Preconditions (where relevant)

4.1 curriculum-related	• Software Engineering
4.2 competence - related	• Programing Experience

5. Facilities and equipment (where relevant)

5.1 for the course	• Overhead Projector
5.2 for the laboratory/project	• 20 PC

6. Specific competences acquired

Professional competences	Work effectively in interdisciplinary contexts, in particular to bridge the gap between computing technology and the clients business and to interpret and respect extra-technical constraints deriving from the business organization
Transversal competences	Understand and be able to use specific tools, components, and frameworks and also abstract elements such as algorithms and architectures Organize and lead development teams, including team-building and negotiation Serve as an agent of change for introducing new technology

7. Course objectives (as resulting fro n the grid of specific competences)

7.1 Subject general goal	The course is designed to provide detailed insight into the management methods and responsibilities involved in project management specific to software development. The course is dedicated to students who want to develop specific skills, styles and approaches in this area. The laboratory performs practical examples according to (and synchronized) with the material taught at the course. In addition to being able to analyze homework issues, the lab offers the opportunity for students to work in an organized setting, to a joint project, in teams formed at random, and to contribute to interactive debates on different approaches in the structure of some teams and / or in the design / implementation technique of certain projects.
7.2 Specific objectives	Through this discipline it is desired to understand clearly the problems, the success factors and the risks associated with the development of the projects in the software field, the details of the stages and processes within the life cycle of a project as well as the planning and management techniques of a project software.

8. Content

8. 1 Course	Teaching Methods	Observations
The course as a whole • Introduction • Project management • Fundamental mistakes in addressing projects and ways to identify them • Interactive discussion on wrong approaches	Teaching face to face with video projector. Presentation of Internet articles / specialty magazines. Presentations of students at the course.	
Overview of Project Management • Organizational structures • Project organizational plans • Interactive discussion on a number of organizational models applied in various companies		
Planning phase • Life cycle development models • Choice of lifecycle models for projects • Interactive discussion on a series of lifecycle models, the choice of 2 camps that will present each distinct variant and will fight against the "opponents" variant while supporting their own variants.		
Estimations and Budget • Estimates, Budget, Selection of Projects • Investment recovery models		

• Interactive discussion on a range of budgeting models		
Project Planning • Flow of project flow • UML basics. Types of Charts • Interactive development of a UML class diagram		
Risk and change management • Risk management • Change control		
Development Management • Team models • Conflict management and motivation of individuals • Interactive discussion about choosing a case study		
Project control • Stage reporting • Project metrics • Advanced UML notions • Interactive discussion about UML detailing for the case study chosen		
Process testing systems • Test specifications • Testing tools • Interactive discussion of the description of UML for the case study chosen, comparisons with other approaches from similar projects; the choice of 2 camps that will present each distinct variant and will fight the variant of "opponents", while supporting their own variant.		
The final stages of the projects • Recovery of projects • documentation • Migration • Post-project evaluation • Closing the project • Interactive discussion on the description and implementation of UML for the chosen case study, comparisons with other approaches from similar projects; the choice of 2 camps that will present each distinct variant and will fight the variant of "opponents", while supporting their own variant.		
Success of the project • Management of project support services • Expectations • Metrics of success • Interactive discussion of expectations and measurement of case study success		

Bibliography

"Rapid Development", McConnell, Steve, Microsoft Press, 1996, ISBN 1-55615-900-5.
"Information Technology Project Management", Schwalbe, Kathy, 2nd ed., Course Technology, 2002, ISBN 0-619-03528-5.
"UML Distilled: A Brief Guide to the Standard Object Modeling Language", Fowler, Martin. 3rd ed., Addison-Wesley. ISBN 0-321-19368-7.

8. 2 Seminar	**Teaching Methods**	**Observations**
8.3 Laboratory		

The laboratory as a whole • Introduction • Presentation of a case study • Overview of Project Management • The organizational structure of the case study • Planning phase • Interactive choice of a life model for the case study • Estimates and Budget • Choosing a budgeting model • Project planning • Flow of project flow • Case study diagrams • Risk and change management • Risk management • Risks for the case study	Individual, theoretical and practical activity;. design and teamwork problems, low and medium complexity	
Bibliography Same as the course		

9. Subject's relevance to the epistemic community, professional associations and representative employers in fields significant for the program

- Project Management is a very important part of any software development enterprise. The course prepare the students with the necessary skills and knowledge for the positions of team leaders, project managers, senior managers. etc.

10. Evaluation

Activity	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Weight in final grade
10.4 Course	Problem solving correctness	Written exam	40%
	Presentation of scientifically papers	Oral evaluation	15% (extra)
10.5 Seminar			
10.5 Laboratory/Project	Project development	Project work during semester	20%
	Project success	Final project evaluation, normalized using the team and personal marks	25%
10.7 Minimal standard of performance			
• **Minimum 50% of the marks from Seminaries Assignments and Course Activities part (3 points out of 6)** • **Minimum of 50 % from final examination (2 points out of 4)**			

Agent-Oriented Software Engineering

1. Information about the program

1.1 Higher education institution	University POLITEHNICA of Bucharest
1.2 Faculty	Faculty of Engineering in Foreign Languages
1.3 Department	Department of Engineering in Foreign Languages
1.4 Field of study	Computers and Information Technology
1.5 Study cycle	Master
1.6 Program / Qualification	Software Engineering

2. Data about the subject

2.1 Name of subject			Agent-Oriented Software Engineering				
2.2 Course holder							
2.3 Seminar holder							
2.4 Laboratory/project holder							
2.5 Year of study	6	2.6 Semester	2	2.7 Evaluation type	E	2.8 Subject type	DAP

3. Estimated time (hours per semester) of didactic activities

3.1 Number of hours per week	4	course hours	2	seminar	1	laboratory	1
3.2. Number of hours per semester	56	course hours	28	seminar	14	laboratory	14
3.3.Distribution of spend time:							**h.**
Study of textbooks, bibliography and course notes							
Supplementary study in library, on electronic platforms, on the fieldwork							
Preparation of seminars/laboratories, home assignments, papers, portfolios, essays							
Tutoring							
Examinations							
Other activities							

3.4 Total hours of individual study	**50**	
3.5 Total hours per semester [1]	**106**	
3.6 Number of credits	**4**	

4. Preconditions (where relevant)

4.1 curriculum- related	•
4.2 competence - related	•

[1] Numărul total de ore nu trebuie să depăşească valoarea (Număr credite) x 27 ore

5. Facilities and equipment (where relevant)

5.1 for the course	•
5.2 for the course seminar	•
5.3 for the laboratory/project	•

6. Specific competences acquired

Professional competences	• Apply design and development methods and techniques as appropriate to realize solutions along the whole life-cycle of the software product
Transversal competences	• Understand and be able to use specific tools, components, and frameworks and also abstract elements such as algorithms and architectures • Organize and lead development teams, including team-building and negotiation • Serve as an agent of change for introducing new technology

7. Course objectives (as resulting from the grid of specific competences)

7.1 Subject general goal	• Acquiring theoretical and practical knowledge about intelligent agents and multi-agent systems • Studying agent types and multi-agent types • Presenting the reasoning methods for intelligent agents • Presenting the distributed planning methods for MAS, the coordination mechanisms and elements of agent oriented programming • Studying development methods for applications based on the multi-agent paradigm
7.2 Specific objectives	• Developing applications based on intelligent agents • Abilities for the design and development of multi-agent systems

8. Content

8.1 Course	**Teaching methods**	**Observations**
Introduction • Agents and Multi-Agent Systems • Cognitive and Reactive Agent	Face to face teaching	Bibliographic materials are used.

Architectures • Languages and communication protocols for MAS Coordination • Coordination in solving tasks • Distributed planning in MAS • Negotiating techniques and protocols Specific Applications • Agent oriented programming MAS Platforms • Multi agent systems applications		
Bibliography • M. Wooldridge. An Introduction to Multiagent Systems. John Wiley and Sons, 2002. • Bordini, Rafael H., Jomi Fred Hübner, and Michael Wooldridge. Programming multi-agent systems in AgentSpeak using Jason. Vol. 8. John Wiley & Sons, 2007. • G. Weiss (ed.). Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press, 2000. . – disponibila in biblioteca Laboratorului AI-MAS • L. Padgham, M. Winikoff . Developing Intelligent Agent Systems: A Practical Guide. Wiley Series in Agent Technology, 2004. – disponibila in biblioteca Laboratorului AI-MAS • F. L. Bellifemine, G. Caire, D. Greenwood. Developing Multi-Agent Systems with JADE, Wiley Series in Agent Technology, 2007. – disponibila in biblioteca Laboratorului AI-MAS • A. M. Florea. Sisteme multi-agent. Note de curs – format electronic, in curs de redactare pentru publicare. • A.M. Florea. Multi-agent Systems. Slides for the CS525 taught at Worchester Polytechnic Institute, Massachusetts, USA		
8.2 Seminar	**Teaching methods**	**Observations**
• Implementing a multi-agent system using the BDI architecture • Implementing the environment for multi-agent systems	Face to face teaching	
8.3 Laboratory		
• programming agents using Jason and Cartago programming languages	Face to face teaching	
Bibliography • M. Wooldridge. An Introduction to Multiagent Systems. John Wiley and Sons, 2002 • Bordini, Rafael H., Jomi Fred Hübner, and Michael Wooldridge. Programming multi-agent systems in AgentSpeak using Jason. Vol. 8. John Wiley & Sons, 2007. • G. Weiss (ed.). Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press, 2000. . – disponibila in biblioteca Laboratorului AI-MAS • L. Padgham, M. Winikoff . Developing Intelligent Agent Systems: A Practical Guide. Wiley Series in Agent Technology, 2004. – disponibila in biblioteca Laboratorului AI-MAS • F. L. Bellifemine, G. Caire, D. Greenwood. Developing Multi-Agent Systems with JADE, Wiley Series in Agent Technology, 2007. – disponibila in biblioteca Laboratorului AI-MAS • A. M. Florea. Sisteme multi-agent. Note de curs – format electronic, in curs de redactare pentru publicare. • A.M. Florea. Multi-agent Systems. Slides for the CS525 taught at Worchester Polytechnic Institute, Massachusetts, USA		

9. Subject's relevance to the epistemic community, professional associations and representative employers in fields significant for the program

•

10. Assessment

Activity	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Weight in final grade
10.4 Course	The quality of the provided solution	Written exam	20
	The quality of the provided solution	Written exam	60
10.5 Seminar	The quality of the provided solution	Solutions evaluation	10
10.6 Laboratory/Project	Laboratory activity	Oral evaluation	5
	The quality of the provided solution	Solutions evaluation	5
10.7 Minimal standard of performance			
• Obtaining minimum 50% of the final exam • Attending at least 5 laboratory sessions			

Special Topics in Software Engineering

1. Information about the program

1.1 Higher education institution	University POLITEHNICA of Bucharest
1.2 Faculty	Faculty of Engineering in Foreign Languages
1.3 Department	Department of Engineering in Foreign Languages
1.4 Field of study	Computers and Information Technology
1.5 Study cycle	Master
1.6 Program / Qualification	Software Engineering

2. Data about the subject

2.1 Name of subject			Special Topics in Software Engineering				
2.2 Course holder							
2.3 Seminar holder							
2.4 Laboratory/project holder							
2.5 Year of study	2	2.6 Semester	1	2.7 Evaluation type	E	2.8 Subject type	DO

3. Estimated time (hours per semester) of didactic activities

3.1 Number of hours per week	2	course hours	1	seminar		laboratory	1
3.2. Number of hours per semester	28	course hours	14	seminar		laboratory	14
3.3.Distribution of spend time:							**h.**
Study of textbooks, bibliography and course notes							14
Supplementary study in library, on electronic platforms, on the fieldwork							10
Preparation of seminars/laboratories, home assignments, papers, portfolios, essays							14
Tutoring							2
Examinations							2
Other activities							

3.4 Total hours of individual study	**42**	
3.5 Total hours per semester [1]	**70**	
3.6 Number of credits	**3**	

4. Preconditions (where relevant)

4.1 curriculum- related	• Programming Languages, Computer Networks
4.2 competence - related	•

5. Facilities and equipment (where relevant)

5.1 for the course	• Projector, blackboard/whiteboard
5.2 for the course seminar	•
5.3 for the laboratory/project	• Laboratory with computers • Internet connection • Development boards with sensors and communication

[1] Numărul total de ore nu trebuie să depășească valoarea (Număr credite) x 27 ore

	capabilities

6. Specific competences acquired

Professional competences	Design software solutions using responsible engineering aproaches. Apply theories and models in the design process
Transversal competences	Understand and be able to use specific tools, components, and frameworks and also abstract elements such as algorithms and architectures. Organize and lead development teams, including team-building and negotiation Serve as an agent of change for introducing new technology

7. Course objectives (as resulting from the grid of specific competences)

7.1 Subject general goal	• Study the Internet of Things paradigm, using intelligent devices in complex information systems
7.2 Specific objectives	• Knowing the Internet of Things paradigm • Learning specific protocols for machine-to-machine communication • Developing applications on development boards using sensors and actuators • Connecting smart devices to the Internet

8. Content

8.1 Course	**Teaching methods**	**Observations**
Introduction to IOT, Networks, Taxonomy, Examples	Blackboard, projector, Moodle	1
IOT systems architecture		1
Development boards, Capabilities, Programming		2
Analog and digital sensors		1
Actuators		1
Serial communication protocols		1
Machine-to-machine communication in wireless networks		1
Low-energy communication		1
Using cloud services		1
Connecting to smartphone applications		2
Digital signal processing on development boards		2
Bibliography		
8.2 Seminar	**Teaching methods**	**Observations**
	Moodle, individual work at	

8.3 Laboratory	computer	
Programming Arduino (or similar) development boards		**2**
Programming Raspberry Pi (or similar) development boards		**2**
Reading digital and analog sensor data		**2**
Serial communication (SPI, I2C)		**1**

Bibliography

- „Collaborative Internet of Things (C-IoT): for Future Smart Connected Life and Business", Fawzi Behmann, Kwok Wu, ISBN: 978-1-118-91374-1, 2015
- The Internet of Things: Key Applications and Protocols, 2nd Edition, Olivier Hersent, David Boswarthick, Omar Elloumi, ISBN: 978-1-119-99435-0, 2012
- „Professional Android Sensor Programming", Greg Milette, Adam Stroud, ISBN: 978-1-118-18348-9, 2012
- „Raspberry Pi 3: Beginner to Pro – Step by Step Guide", Timothy Short, 2016
- „Beginning Sensor Networks with Arduino and Raspberry Pi", Charles Bell, ISBN-13: 978-1430258247, 2013

9. Subject's relevance to the epistemic community, professional associations and representative employers in fields significant for the program

- The course and laboratory were prepared after extensive study of similar programs offered by prestigious universities and adapted to be integrated in the current study program.

10. Assessment

Activity	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Weight in final grade
10.4 Course	Knowing the theory	Written exam	40
	Solving a programming task		
10.5 Seminar			
10.6 Laboratory/Project	Attendance + homework + activity	Oral examination	30
	Programming tests	Written test during the laboratory	30
10.7 Minimal standard of performance			

- Knowing basic theory (IOT architectures, fundamental protocols)
- Designing of IOT systems for satisfying requirements presented in natural language.

Software Testing

1. Information about the program

1.1 Higher education institution	University POLITEHNICA of Bucharest
1.2 Faculty	Faculty of Engineering in Foreign Languages
1.3 Department	Department of Engineering in Foreign Languages
1.4 Field of study	Computers and Information Technology
1.5 Study cycle	Master
1.6 Program / Qualification	Software Engineering

2. Data about the subject

2.1 Name of subject			Software Testing				
2.2 Course holder							
2.3 Seminar holder							
2.4 Laboratory/project holder							
2.5 Year of study	1	2.6 Semester	1	2.7 Evaluation type	C	2.8 Subject type	DPA/DO

3. Estimated time (hours per semester) of didactic activities

3.1 Number of hours per week	3	course hours	2	seminar	0	laboratory	1
3.2. Number of hours per semester	42	course hours	14	seminar	0	laboratory	14
3.3.Distribution of spend time:							**h.**
Study of textbooks, bibliography and course notes						20	
Supplementary study in library, on electronic platforms, on the fieldwork						20	
Preparation of seminars/laboratories, home assignments, papers, portfolios, essays						10	
Tutoring						3	
Examinations						3	
Other activities							

3.4 Total hours of individual study	**66**	
3.5 Total hours per semester [1]	**108**	
3.6 Number of credits	**4**	

4. Preconditions (where relevant)

4.1 curriculum- related	• Introduction to Information Technology • Data Structures and Algorithms
4.2 competence - related	•

[1] Numărul total de ore nu trebuie să depășească valoarea (Număr credite) x 27 ore

5. Facilities and equipment (where relevant)

5.1 for the course	• Overhead Projector
5.2 for the course seminar	•
5.3 for the laboratory/project	• 20 PC

6. Specific competences acquired

Professional competences	• Apply design and development methods and techniques as appropriate to realize solutions along the whole life-cycle of the software product
	• Understand and be able to use specific tools, components, and frameworks and also abstract elements such as algorithms and architectures • Organize and lead development teams, including team-building and negotiation • Serve as an agent of change for introducing new technology

7. Course objectives (as resulting from the grid of specific competences)

7.1 Subject general goal	• Scientific foundations for software engineering depend on the use of precise testing methodology for checking the correctness of a software under test. This course aims to bring the student the information regarding rigorous testing starting with test planning, functional and non-functional testing strategies, conformance standardisation testing, automatic test generation, test reporting and debugging, etc.
7.2 Specific objectives	• How to use formal specification methods in software development for checking the correctness of a specification • How to develop test planning and do test reporting • How to use different strategies for effective testing.

8. Content

Course	Teaching methods	Observations
• Background knowledge • Use of probabilities in testing • Design phases: Testing the design – Formal methods • Planning for testing • Static testing: Audits, Interviews • Detailed Design and Coding, Transfer and Maintanance phases: Main Testing Methodologies • Non-functional Testing • Automtic test derivation • Standardisation testing conformance • Reporting the results	Lecturing	

Bibliography

- Elaine A. Rich, Automata, Computability and Complexity: Theory and ApplicationsSep 28, 2007
- John E. Hopcroft and Rajeev Motwani, Introduction to Automata Theory, Languages, and Computation, 2006
- "Programare Functionala, O perspectiva pragmatica", C. Giumale, Editura tehnica, 1997
- "Z An Itroduction to Formal Methods", A. Diller, John Wiley & Sons, 1994
- Rex Black. Pragmatic Software Testing: Becoming an Effective and Efficient Test Professional. John Wiley & Sons, 2007.
- Rex Black. Testing Metrics. RBCS, 2012

Rex Black, et al. Foundations of Software Testing, 3rd Edition. Thomson Learning, 2011.

8.2 Seminar	Teaching Method	Observation

8.3 Laboratory		
Introduction to Testing	Laboratory Work	
Testing the design with Prommela And Spin	Laboratory Work	
Elaboration of a testing Plan	Laboratory Work	
Testing plan implementation	Laboratory Work	
Reporting Testing results	Laboratory Work	

Bibliography

- Elaine A. Rich, Automata, Computability and Complexity: Theory and ApplicationsSep 28, 2007
- John E. Hopcroft and Rajeev Motwani, Introduction to Automata Theory, Languages, and Computation, 2006
- "Programare Functionala, O perspectiva pragmatica", C. Giumale, Editura tehnica, 1997
- "Z An Itroduction to Formal Methods", A. Diller, John Wiley & Sons, 1994
- Rex Black. Pragmatic Software Testing: Becoming an Effective and Efficient Test Professional. John Wiley & Sons, 2007.
- Rex Black. Testing Metrics. RBCS, 2012
- Rex Black, et al. Foundations of Software Testing, 3rd Edition. Thomson Learning, 2011.

9. Subject's relevance to the epistemic community, professional associations and representative employers in fields significant for the program

- Testing is a key piece in the development of a correct software. It is estimated that between 50-80% of the software development time goes in testing. Course helps students to acquire good testing skills such that they will be able to cope well with testing activities as future engineers and managers.

10. Assessment

Activity	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Weight in final grade
10.4 Course	Course Presence and Activities	Presence and Activities Evaluation	10%
	Final Examinations	Written Exam	40%
10.5 Seminar			
10.6 Laboratory/Project	Laboratory Assignment	Assignments Correction	50%
10.7 Minimal standard of performance			

- **Minimum 50% of the marks from Seminaries Assignments and Course Activities part (3 points out of 6)**
- **Minimum of 50 % from final examination (2 points out of 4)**