SYLLABUS

1. Data about the program of study

1.1	Institution	The Technical University of Cluj-Napoca
1.2	Faculty	Automation and Computer Science
1.3	Department	Computer Science
1.4	Field of study	Computer Science and Information Technology
1.5	Cycle of study	Bachelor of Science
1.6	Program of study/Qualification	Computer Science/ Engineer
1.7	Form of education	Full time
1.8	Subject code	23.

2. Data about the subject

2.1	Subject name					Systems Theory					
2.2	Subject area					Computer Science and Information Technology					
2.3	Course responsible/lecturer					Assoc. prof. dr. eng. Paula Raica – Paula.Raica@aut.utcluj.ro					
2.4	Teachers in charge of applications					Conf.dr.ing. Paula Raica, Sl.dr.ing. Iulia Clitan, Asist.dr.ing. Alexandru Codrean, Ing. Zoltan Nagy					
2.5	Year of study	II	2.6	Semester	4	2.7	Assessment	exam	2.8	Subject category	DID/OB

3. Estimated total time

Sem.	Subject name	Lecture	Applications			Lecture	Applications			Individual study	TOTAL	Credit
		[hours / week.]				[hours / semester]						
			S	L	P		S	L	P			
4	**Systems Theory**	**2**	**-**	**2**	**-**	**28**	**-**	**28**	**-**	**48**	**104**	**4**

3.1	Number of hours per week	4	3.2	of which, course	2	3.3	applications	2
3.4	Total hours in the teaching plan	56	3.5	of which, course	28	3.6	applications	28
Individual study								Hours
Manual, lecture material and notes, bibliography								20
Supplementary study in the library, online and in the field								5
Preparation for seminars/laboratory works, homework, reports, portfolios, essays								20
Tutoring								
Exams and tests								3
Other activities								

3.7	Total hours of individual study	48
3.8	Total hours per semester	104
3.9	Number of credit points	4

4. Pre-requisites (where appropriate)

4.1	Curriculum	Mathematical Analysis_II (Integral calculus and differential equations, Linear algebra
4.2	Competence	Differential equations, complex numbers, Laplace transform, linear algebra

5. Requirements (where appropriate)

5.1	For the course	N/A
5.2	For the applications	Reading and understanding of the lecture notes.

6. Specific competences

Professional competences	**C1 –** Operating with basic Mathematical, Engineering and Computer Science concepts (4 credits) **C1.1 –** Recognizing and describing concepts that are specific to the fields of calculability, complexity, programming paradigms, and modeling computational and communication systems **C1.2 –** Using specific theories and tools (algorithms, schemes, models, protocols, etc.) for explaining the structure and the functioning of hardware, software and communication systems **C1.3 –** Building models for various components of computing systems **C1.4 –** Formal evaluation of the functional and non-functional characteristics of computing systems **C1.5 –** Providing a theoretical background for the characteristics of the designed systems
Cross competences	N/A

7. Discipline objectives (as results from the *key competences gained*)

7.1	General objective	The general objective of the course is to introduce the fundamental principles of linear system modeling, analysis and feedback control and to evaluate feedback control systems with desired behavior.
7.2	Specific objectives	The specific objectives are to acquire the knowledge and techniques related to: - mathematical system modeling (differential equations, input-output representation as transfer functions, block diagrams) for simple applications - linear system analysis (assessment of stability and performance properties of linear systems) in time and frequency domains - design of feedback controllers such as PID, lead and lag compensators for linear systems using s-domain techniques - linear sampled-data system representation and analysis

8. Contents

8.1. Lecture (syllabus)		Teaching methods	Notes
1	Introduction to systems theory and control engineering. Intoduction to system modeling. Linear approximation.	Lecture, visual presentations, demonstrations	N/A
2	Input/output models. System response. State-space models.		
3	Conversion between transfer function and state space. Block diagrams.		
4	Linear system analysis. 1st and 2nd order systems. Steady-state error.		
5	Higher order systems. Dominant poles. Stability of linear continuous systems.		
6	System analysis using root locus.		
7	Frequency response. Bode diagrams.		
8	Controller design. Lead-lag compensation.		
9	System analysis. Applications. Midterm exam.		
10	PID – the basic technique for feedback control.		
11	Controlability. Observability. State feedback.		
12	Sampled-data systems.		
13	Digital control systems		
14	Controller design – aplications. Sampled-data systems – applications.		

Bibliography
1. R. C. Dorf, R. Bishop, “Modern Control Systems”, Addison-Wesley, 2004;
2. K. Ogata , “Modern Control Engineering”, Prentice Hall, 1990.
3. K. Dutton, S. Thompson, B. Barraclough, “The Art of Control Engineering”, Addison-Wesley, 1997
4. William S. Levine (editor), “The Control Handbook”, CRC Press and IEEE Press, 1996
5. Lecture notes available on the course webpage: http://rocon.utcluj.ro/st

8.2. Applications (Seminars, Laboratory, Projects)		Teaching methods	Notes
1	Introduction to Matlab. Simulation of dynamical systems	Class	4 hours

2	Linear approximation of differential equations. Transfer functions. System response.	discussion, Supervised exercise solving using Matlab Miniprojects – individul student reports	4 hours
3	Block diagram models. 1st and 2nd order system analysis. Steady-state error		4 hours
4	System stability. Root locus		4 hours
5	Frequency response. Bode diagrams		4 hours
6	Lead-lag compensation. PID controllers		4 hours
7	State feedback. Sampled-data systems.		4 hours

Bibliography

1. Paula Raica, “Control Engineering. Exercises”, Editura Mediamira, 2001
2. Lecture notes available on the course webpage: http://rocon.utcluj.ro/st

9. Bridging course contents with the expectations of the representatives of the community, professional associations and employers in the field

The course content combines theoretical knowledge with applications and focuses on the formulation and solution of specific problems that may occur in various engineering fields. Application of the control theory concepts are specific to most of the engineering disciplines. The course level is introductory and the intent is to motivate and prepare students for further study in related areas and to conduct projects in real-life applications.

10. Evaluation

Activity type	10.1	Assessment criteria	10.2	Assessment methods	10.3	Weight in the final grade
Course		Ability to solve exercises related to linear system modeling and analysis		Midterm exam – writen examination		40%
		Ability to solve exercises related to system design and analysis of sampled-data systems		Final exam - writen examination		60%
Applications		Answer simple questions from the topic of the lab applications		Lab tests (optional)		30% (optional, but may contribute to a higher grade)

10.4 Minimum standard of performance

Solution of simple exercises applying the knowledge and techniques presented in the course.

40% Midterm grade + 60%Final grade + 30%Lab grade > 5

Course responsible
Conf.dr.ing. Paula Raica

Head of department
Prof.dr.eng. Rodica Potolea

SYLLABUS

1. Data about the program of study

1.1	Institution	The Technical University of Cluj-Napoca
1.2	Faculty	Automation and Computer Science
1.3	Department	Computer Science
1.4	Field of study	Computer Science and Information Technology
1.5	Cycle of study	Bachelor of Science
1.6	Program of study/Qualification	Computer Science/ Engineer
1.7	Form of education	Full time
1.8	Subject code	24.

2. Data about the subject

2.1	Subject name					Computer Architecture					
2.2	Subject area					Computer Science and Information Technology					
2.3	Course responsible/lecturer					S.l.dr.ing. Mihai Negru – Mihai.Negru@cs.utcluj.ro					
2.4	Teachers in charge of applications					Conf.dr. ing. Florin Oniga, S.l.dr.ing. Mihai Negru, { Florin.Oniga, Mihai.Negru }@cs.utcluj.ro					
2.5	Year of study	II	2.6	Semester	4	2.7	Assessment	exam	2.8	Subject category	DID/OB

3. Estimated total time

Sem.	Subject name	Lecture	Applications			Lecture	Applications			Individual study	TOTAL	Credit
		[hours / week.]				[hours / semester]						
			S	L	P		S	L	P			
4	Computer Architecture	**2**	**-**	**2**	**-**	**28**	**-**	**28**	**-**	**74**	**130**	**5**

3.1	Number of hours per week	4	3.2	of which, course	2	3.3	applications	2
3.4	Total hours in the teaching plan	56	3.5	of which, course	28	3.6	applications	28
Individual study								Hours
Manual, lecture material and notes, bibliography								28
Supplementary study in the library, online and in the field								14
Preparation for seminars/laboratory works, homework, reports, portfolios, essays								28
Tutoring								0
Exams and tests								4
Other activities								0

3.7	Total hours of individual study	74
3.8	Total hours per semester	130
3.9	Number of credit points	5

4. Pre-requisites (where appropriate)

4.1	Curriculum	1. Logic design 2. Digital system design
4.2	Competence	Ability to design digital circuits and to implement them in VHDL

5. Requirements (where appropriate)

5.1	For the course	blackboard, video projector, laptop
5.2	For the applications	desktop/laptop computer, Xilinx ISE / VIVADO, FPGA development boards

6. Specific competences

Professional competences	**C2 –** Designing hardware, software and communication components (5 credits) **C2.1 –** Describing the structure and functioning of computational, communication and software components and systems **C2.2 –** Explaining the role, interaction and functioning of hardware, software and communication components **C2.3 –** Building the hardware and software components of some computing systems using algorithms, design methods, protocols, languages, data structures, and technologies **C2.4 –** Evaluating the functional and non-functional characteristics of the computing systems using specific metrics **C2.5 –** Implementing hardware, software and communication systems
Cross competenc	N/A

7. Discipline objectives (as results from the *key competences gained*)

7.1	General objective	Knowing and understanding the concepts of organization and functioning for central processing units, memories, input/output, and using these concepts for design.
7.2	Specific objectives	• Applying methods for representation and design at system level for digital circuits • Instruction Set Architecture (ISA) specification • Writing simple programs in assembly languages and machine code • Specification, design, implementation, and testing of Central Processing Units (CPU) – micro architecture – data path – command units • Understanding memory organization and I/O operations • Understanding modern trends in computer architectures

8. Contents

8.1. Lecture (syllabus)		**Teaching methods**	**Notes**
1	Introduction	Oral presentation backed up by multimedia equipment, interactive communication, blackboard problem solving	
2	High-Level Synthesis		
3	Instruction Set Architecture (ISA)		
4	CPU Design - Single Cycle CPU		
5	Computer Arithmetic and Simple Arithmetic Logic Units		
6	CPU Design - Multi Cycle CPU Data path		
7	CPU Design - Multi Cycle CPU Control		
8	CPU Design – Pipelined CPU		
9	Advanced Pipelining – Static and Dynamic Scheduling of the Execution		
10	Branch Prediction		
11	Superscalar Architectures		
12	Memory		
13	I/O and Interconnection Structures		
14	Problem solving		

Bibliography

1. D. A. Patterson, J. L. Hennessy, “Computer Organization and Design: The Hardware/Software Interface”,5th edition, ed. Morgan–Kaufmann, 2013.
2. D. A. Patterson and J. L. Hennessy, “Computer Organization and Design: A Quantitative Approach”,5th edition, ed. Morgan-Kaufmann, 2011.
3. Vincent P. Heuring, et al., “Computer Systems Design and Architecture”, Addison-Wesley, USA, 1997.
4. A. Tanenbaum, “Structured Computer Organization”, Prentice Hall, USA, 1999.
5. MIPS32 Architecture for Programmers, Volume I: “Introduction to the MIPS 32™ Architecture”.
6. MIPS32 Architecture for Programmers, Volume II: “The MIPS 32™ Instruction Set”.

Online bibliography
M. Negru, F. Oniga, S. Nedevschi, Lecture slides http://users.utcluj.ro/~negrum

8.2. Applications (Laboratory)		Teaching methods	Notes
1	Introduction in the Xilinx ISE environment and the FPGA development board	Blackboard quick overview of key issues, exercises, experimenting with FPGA development boards with specialized IDEs for circuit design and implementation (Xilinx ISE)	
2	Design and Implementation of Combinational CPU Components		
3	Design and Implementation of Sequential CPU Components		
4	Design of a Single Cycle CPU 1 (MIPS)		
5	Design of a Single Cycle CPU 2 (MIPS)		
6	Design of a Single Cycle CPU 3 (MIPS)		
7	Design of a Single Cycle CPU 4 (MIPS)		
8	Midterm practical evaluation on the FPGA board		
9	Pipelined CPU Design		
10	Pipelined CPU Design		
11	Pipelined CPU Design		
12	Pipelined CPU interfacing		
13	Practical evaluation of the pipelined CPU on the FPGA board		
14	Final Tests and Evaluation		

Bibliography
Online bibliography
M. Negru, F. Oniga, S. Nedevschi, Laboratory guide http://users.utcluj.ro/~negrum

9. Bridging course contents with the expectations of the representatives of the community, professional associations and employers in the field

Computer Architecture is one of the fundamental subjects of the Computer Science and Information Technology field. It combines fundamental and practical aspects used for digital circuits design and implementation. The content of this subject is harmonized with the specific curricula of other national and international universities, and is evaluated by the Romanian government agencies (CNEAA and ARACIS). The practical aspects involve getting familiar with and using development products and tools provided by companies from Romania, Europe, and USA (ex. Xilinx, Digilent).

10. Evaluation

Activity type	10.1	Assessment criteria	10.2	Assessment methods	10.3	Weight in the final grade
Course		Testing the theoretical knowledge, the ability of problem solving, presence and activity		Written exam		50 %
Applications		Practical ability to solve and implement specific problems related to processor design, presence and activity		Lab exam, periodical assessment of results		50 %
10.4 Minimum standard of performance						
Knowing the fundamental theory of the subject, the ability to design and implement a processor with a reduced set of instructions.						

Course responsible
S.l.dr.ing. Mihai Negru

Head of department
Prof.dr.eng. Rodica Potolea

SYLLABUS

1. Data about the program of study

1.1	Institution	The Technical University of Cluj-Napoca
1.2	Faculty	Automation and Computer Science
1.3	Department	Computer Science
1.4	Field of study	Computer Science and Information Technology
1.5	Cycle of study	Bachelor of Science
1.6	Program of study/Qualification	Computer Science/ Engineer
1.7	Form of education	Full time
1.8	Subject code	25.

2. Data about the subject

2.1	Subject name					Fundamental Algorithms					
2.2	Subject area					Computer Science and Information Technology					
2.3	Course responsible/lecturer					Prof. dr. eng. Rodica Potolea – Rodica.Potolea@cs.utcluj.ro					
2.4	Teachers in charge of applications					S.l.dr.eng. Camelia Lemnaru – Camelia.Lemnaru@cs.utcluj.ro					
2.5	Year of study	II	2.6	Semester	4	2.7	Assessment	exam	2.8	Subject category	DID/OB

3. Estimated total time

Sem.	Subject name	Lecture	Applications			Lecture	Applications			Individual study	TOTAL	Credit
		[hours / week.]				[hours / semester]						
			S	L	P		S	L	P			
4	**Fundamental Algorithms**	**2**	**1**	**2**	**-**	**28**	**14**	**28**	**-**	**60**	**130**	**5**

3.1	Number of hours per week	5	3.2	of which, course	2	3.3	applications	3
3.4	Total hours in the teaching plan	70	3.5	of which, course	28	3.6	applications	42

Individual study	Hours
Manual, lecture material and notes, bibliography	21
Supplementary study in the library, online and in the field	16
Preparation for seminars/laboratory works, homework, reports, portfolios, essays	16
Tutoring	8
Exams and tests	9
Other activities	

3.7	Total hours of individual study	60
3.8	Total hours per semester	130
3.9	Number of credit points	5

4. Pre-requisites (where appropriate)

4.1	Curriculum	Imperative programming languages (C şi / sau Java) Data Structures and Algorithms
4.2	Competence	Acquire the abilities of designing, implementing, testing and evaluating programs to solve specific problems

5. Requirements (where appropriate)

5.1	For the course	Whiteboard, projector, computer
5.2	For the applications	Computers/Network of computers, C ++

6. Specific competences

Professional competences	**C3.** Problems solving using specific Computer Science and Computer Engineering tools (5 credit points) **C3.1-** Identifying classes of problems and solving methods that are specific to computing systems **C3.2 -** Using interdisciplinary knowledge, solution patterns and tools, making experiments and interpreting their results **C3.3 -** Applying solution patterns using specific engineering tools and mehods **C3.4 -** Evaluating, comparatively and experimentally, the available alternative solutions for performance optimization **C3.5 -** Developing and implementing informatic solutions for concrete problems **C4.** Improving performances of hardware, software and communication systems **C4.1 -** Identifying and describing the defining performance elements of hardware, software and communication systems **C4.2 -** Explaining the interaction of the factors that determine the performances of hardware, software and communication systems **C4.3 -** Applying fundamental methods and principles for increasing performance of hardware, software and communication systems **C4.4 -** Choosing criteria and methods for performance evaluation of hardware, software and communication systems **C4.5 -** Developing performance based professional solutions for hardware, software and communication systems
Cross competences	N/A

7. Discipline objectives (as results from the *key competences gained*)

7.1	General objective	• Acquiring modern study of algorithms: design and analysis
7.2	Specific objectives	• Learn to identify and design efficient solutions to problems • Learn methods to evaluate efficiency • Learn the basic polynomial algorithms • Learn basic computational complexity • Algorithms description with focus on control structures • Learning the correct implementation following the pseudocode • Efficient implementation of key polynomial algorithms • Estimation of algorithms' efficiency: space and processing time

8. Contents

8.1. Lecture (syllabus)		Teaching methods	Notes
1	Mathematical Foundations: Asymptotical notation, Recurrence	Whiteboard, projector, computer; Lectures, discussions, Q&A sessions	
2	Complexity Classes		
3	Sorting and Order Statistics		
4	Sorting and Order Statistics (continued)		
5	Advanced Data Structures : Hash Tables, Trees		
6	Advanced Data Structures: Heaps, Disjoint Sets		
7	Design and Analysis Advanced Techniques: Dynamic Programming		
8	Design and Analysis Advanced Techniques: Greedy Algorithms		
9	Design and Analysis Advanced Techniques: Damping Analyze		
10	Graphs: Search in a Graph, Minimal Spanning Tree		
11	Graphs: Shortest path		
12	Graphs: Max Flow		
13	Graphs: Bipartite Graphs		
14	Learn the basic Complexity sets and representative problems		

Bibliography

1. T. Cormen, C. Rleiserson, R. Rivest, C. Stein, *Introduction to Algorithms, Second Edition*, The MIT Press, 2001

8.2. Applications (Seminars, Laboratory)		Teaching methods	Notes
1	Efficient implementation and comparison of sorting algorithms	Hands on work on specific algorithms; weekly assessment, feedback, and assistance	
2	Efficient implementation and comparison of sorting algorithms (continued)		
3	Efficient implementation and comparison of lists algorithms		
4	Efficient implementation and comparison of lists algorithms (continued)		
5	Efficient implementation and comparison of trees algorithms		
6	Efficient implementation and comparison of trees algorithms (continued)		
7	Implementation of augmented data structures		
8	Implementation of augmented data structures (continued)		
9	Efficient implementation of graphs algorithms		
10	Efficient implementation of graphs algorithms (continued)		
11	Efficient implementation of graphs algorithms (continued)		
12	Efficient implementation of graphs algorithms (continued)		
13	Approximation algorithms		
14	Final Evaluation		
Bibliography 1. T. Cormen, C. Rleiserson, R. Rivest, C. Stein, *Introduction to Algorithms, Second or third Edition*, The MIT Press, 2001			

9. Bridging course contents with the expectations of the representatives of the community, professional associations and employers in the field

The topic is fundamental in the field of Computer and Information Technology, its content is beyond dispute, familiarizing students with the principles of algorithms design and analysis. The content is similar (including the textbook) with all representative computer science departments in the world, is a core course in the ACM curricula and was rated by the Romanian governmental agencies (CNEAA and ARACIS).

10. Evaluation

Activity type	10.1	Assessment criteria	10.2	Assessment methods	10.3	Weight in the final grade
Course		Theoretical analysis and problem solving skills		Written exam		70% (20% MT + 50% FE)
Applications		Hands on Problem solving skills		Implementation/ hands on		30% (Lab)
10.4 Minimum standard of performance						
Nota ≥5						

Course responsible
Prof.dr.eng. Rodica Potolea

Head of department
Prof.dr.eng. Rodica Potolea

SYLLABUS

1. Data about the program of study

1.1	Institution	The Technical University of Cluj-Napoca
1.2	Faculty	Automation and Computer Science
1.3	Department	Computer Science
1.4	Field of study	Computer Science and Information Technology
1.5	Cycle of study	Bachelor of Science
1.6	Program of study/Qualification	Computer Science/ Engineer
1.7	Form of education	Full time
1.8	Subject code	26.

2. Data about the subject

2.1	Subject name					Fundamental Programming Techniques					
2.2	Subject area					Computer Science and Information Technology					
2.3	Course responsible/lecturer					Prof. dr. eng. Ioan Salomie - Ioan.Salomie@cs.utcluj.ro					
2.4	Teachers in charge of applications					Sl. dr. eng. Tudor Cioară,, Sl. dr. eng. Cristina.Pop, As. Drd. Marcel Antal, As.drd. Claudia Pop, As. Drd. Dorin Moldovan					
2.5	Year of study	II	2.6	Semester	4	2.7	Assessment	exam	2.8	Subject category	DF/OB

3. Estimated total time

Sem.	Subject name	Lecture	Applications			Lecture	Applications			Individual study	TOTAL	Credit
		[hours / week.]				[hours / semester]						
			S	L	P		S	L	P			
4	**Fundamental Programming Techniques**	**2**	**-**	**2**	**-**	**28**	**-**	**28**	**-**	**74**	**130**	**5**

3.1	Number of hours per week	4	3.2	of which, course	28	3.3	applications	28
3.4	Total hours in the teaching plan	56	3.5	of which, course	28	3.6	applications	28

Individual study	Hours
Manual, lecture material and notes, bibliography	18
Supplementary study in the library, online and in the field	16
Preparation for seminars/laboratory works, homework, reports, portfolios, essays	24
Tutoring	
Exams and tests	16
Other activities	

3.7	Total hours of individual study	74
3.8	Total hours per semester	130
3.9	Number of credit points	5

4. Pre-requisites (where appropriate)

4.1	Curriculum	Fundamentals of Object Oriented Programming
4.2	Competence	Knowledge of Object Oriented Programming

5. Requirements (where appropriate)

5.1	For the course	Blackboard, projector, computer, internet
5.2	For the applications	Computers, specific software, internet

6. Specific competences

Professional competences	**C4 -** Improving the performances of the hardware, software and communication systems **C4.1** - Identifying and describing the defining elements of the performances of the hardware, software and communication systems **C4.2** - Explaining the interaction of the factors that determine the performances of the hardware, software and communication systems **C4.3 -** Applying the fundamental methods and principles for increasing the performances of the hardware, software and communication systems **C4.**4 - Choosing the criteria and evaluation methods of the performances of the hardware, software and communication systems **C4.5** - Developing professional solutions for hardware, software and communication systems based on performance optimization
Cross competences	N/A

7. Discipline objectives (as results from the *key competences gained*)

7.1	General objective	Knowledge and using of object-oriented programming techniques for the development of professional software applications
7.2	Specific objectives	-to use programming techniques for designing of classes and interfaces, including contracts and invariants; -to use programming techniques for code reuse by inheritance and polymorphism -to use generic programming techniques for collection processing -to use programming techniques for reflection and event based -to use programming techniques for concurrent and multi-threading programming -to use object-oriented and functional programming in an integrated approach for the development of flexible and efficient programs -to use design patterns and frameworks -to use programming techniques for performance and software maintenance

8. Contents

8.1. Lecture (syllabus)		Teaching methods	Notes
1	Programming techniques with classes and interfaces	-Using modern multimedia teaching methods and direct access to internet; -Challenging questions during lecturers -Students are invited to collaborate in research projects -Personal assistance hours the semester and before the exam	
2	Programming techniques using inheritance and polymorphism		
3	Programming techniques using contracts and invariants		
4	Generic programming techniques		
5	Reflection techniques		
6	Event-driven techniques		
7	Collection programming techniques		
8	Concurrent and multithreading techniques		
9	Flexibility and reuse through design patterns		
10	Main design patterns of type creational, structural and behavioral		
11	Flexibility and reuse through frameworks		
12	Lambda Expressions and Stream processing		
13	Multiparadigm (functional and OO) programming techniques		
14	Programming techniques for efficiency and performance		

Bibliography

1. Ioan Salomie - Tehnici Orientate Obiect, Editura Albastra, Microinformatica, 1995
2. Eric Gamma, Helm, Johnson, Vlissides - Design Patterns, Addison Wesley, 1995 (translated into Romanian

by Teora Publ. as "Sabloane de Proiectare") 3. Joshua Bloch - Effective Java, 2/e Addison Wesley, 2008 4. Ioan Salomie, Note de Curs, http://www.coned.utcluj.ro/~salomie/TP			
8.2. Applications (Seminars, Laboratory, Projects)		Teaching methods	Notes
1	Intro to lab resources and requirements	-Lab sessions with pre-defined exercises and assignments -Using modern multimedia teaching methods and direct access to internet; -Students are invited to collaborate in research projects -Personal assistance hours during the semester and before the exam	
2-3	Assignment 1 - Programming with inheritance and polymorphism		
4-5	Assignment 2 - Programming with contracts (pre and post conditions) and invariants		
6-7	Assignment 3 Programming with multiple threads		
8-9	Assignment 4 – Programming with design patterns		
10-11	Assignment 5 – Programming with generics and Java Collection Framework		
12-13	Assignment 6 – Multi-paradigm programming		
14	Lab Evaluation		
Bibliography - Steve McConnell - Code Complete, 2/e, Microsoft Press, 2004 - http://docs.oracle.com/javase/tutorial/index.html - http://stackoverflow.com/			

9. Bridging course contents with the expectations of the representatives of the community, professional associations and employers in the field

Fundamental Programming Techniques is a subject of the domain "Computers and Information Technology". It teaches students to apply object-oriented programming techniques in designing and implementing of software applications. The content was developed based on the analysis of similar disciplines from other universities as well as based on the requirements of the IT employees. The content was also evaluated by Romanian governmental agencies CNEAA and ARACIS.

10. Evaluation

Activity type	10.1	Assessment criteria	10.2	Assessment methods	10.3	Weight in the final grade
Course		How the students are using programming techniques for: (i) designing of classes and interfaces, including contracts and invariants; (ii) promote code reuse by inheritance and polymorphism; (iii) using generic programming techniques for collection processing; (iv) using programming techniques for concurrent and multi-threading programming; (v) using object-oriented and functional programming in an integrated approach for the development of flexible and efficient programs; (vi) using design patterns and frameworks		written exam		50%
Applications		-Abilities to effectively specify, design, implement and test quality and performance object – oriented programs		-Assessment of programming assignments -Written exam		50%

		-Quality of assessment deliverables -Activity during lab sessions -Presence to lab sessions				
10.4 Minimum standard of performance						
-To be able to use object-oriented programming techniques in designing and implementing software applications -At least mark 5 at the exam and lab evaluation						

Course responsible
Prof.dr.eng. Ioan Salomie

Head of department
Prof.dr.eng. Rodica Potolea

SYLLABUS

1. Data about the program of study

1.1	Institution	The Technical University of Cluj-Napoca
1.2	Faculty	Automation and Computer Science
1.3	Department	Computer Science
1.4	Field of study	Computer Science and Information Technology
1.5	Cycle of study	Bachelor of Science
1.6	Program of study/Qualification	Computer Science/ Engineer
1.7	Form of education	Full time
1.8	Subject code	27.

2. Data about the subject

2.1	Subject name					Operating Systems					
2.2	Subject area					Computer Science and Information Technology					
2.3	Course responsible/lecturer					Conf dr. ing. Adrian Coleşa – adrian.colesa@cs.utcluj.ro					
2.4	Teachers in charge of applications					Conf. dr. ing. Adrian Coleşa – adrian.colesa@cs.utcluj.ro Ing. Radu Ciocas – rciocas@bitdefender.com Ing. Gergo Janos Szeles – jszeles@bitdefender.com Ing. Razvan Teslaru – rteslaru@bitdefender.com Ing. Alexandru Brînduşe – abrinduse@bitdefender.com					
2.5	Year of study	II	2.6	Semester	4	2.7	Assessment	exam	2.8	Subject category	DID/OB

3. Estimated total time

Sem.	Subject name	Lecture	Applications			Lecture	Applications			Individual study	TOTAL	Credit
		[hours / week.]				[hours / semester]						
			S	L	P		S	L	P			
4	**Operating Systems**	**2**	**-**	**2**	**-**	**28**	**-**	**28**	**-**	**74**	**130**	**5**

3.1	Number of hours per week	4	3.2	of which, course	2	3.3	applications	2
3.4	Total hours in the teaching plan	56	3.5	of which, course	28	3.6	applications	28

Individual study	Hours
Manual, lecture material and notes, bibliography	30
Supplementary study in the library, online and in the field	10
Preparation for seminars/laboratory works, homework, reports, portfolios, essays	28
Tutoring	2
Exams and tests	4
Other activities	0

3.7	Total hours of individual study	74
3.8	Total hours per semester	130
3.9	Number of credit points	5

4. Pre-requisites (where appropriate)

4.1	Curriculum	Computer Programming, Data Structures and Algorithms
4.2	Competence	C programming

5. Requirements (where appropriate)

5.1	For the course	Blackboard / Whiteboard, Beamer
5.2	For the applications	Computers, Linux, Windows, Blackboard / Whiteboard

6. Specific competences

Professional competences	**C3:** Problems solving using specific Computer Science and Computer Engineering tools (3 credits) • **C3.1**: Identifying classes of problems and solving methods that are specific to computing systems • **C3.2**: Using interdisciplinary knowledge, solution patterns and tools, making experiments and interpreting their results • **C3.3**: Applying solution patterns using specific engineering tools and methods • **C3.4**: Evaluating, comparatively and experimentally, the available alternative solutions for performance optimization • **C3.5**: Developing and implementing informatic solutions for concrete problems **C4**: Improving the performances of the hardware, software and communication systems (2 credits) • **C4.1**: Identifying and describing the defining elements of the performances of the hardware, software and communication systems • **C4.2**: Explaining the interaction of the factors that determine the performances of the hardware, software and communication systems • **C4.3**: Applying the fundamental methods and principles for increasing the performances of the hardware, software and communication systems • **C4.4**: Choosing the criteria and evaluation methods of the performances of the hardware, software and communication systems • **C4.5**: Developing professional solutions for hardware, software and communication systems based on performance optimization
Cross competences	N/A

7. Discipline objectives (as results from the *key competences gained*)

7.1	General objective	Provide the students a clear understanding of what an OS is, its role and general functionality and the ability to use fundamental system calls of an OS.
7.2	Specific objectives	Let the students: 1. Know and understand the OS specific terminology. 2. Understand the general structure and functionality of an OS. 3. Understand the specific functionality of the most important OS components, like shell, process manager, file system, memory manager, security manager. 4. Understand the functionality of main synchronization mechanisms and be able to use them to solve real synchronization problems. 5. Be able to write C programs to use an OS's (Linux and Windows) system calls.

8. Contents

8.1. Lecture (syllabus)		Teaching methods	Notes
1	**Introduction and basic concepts**. OS's definition, role, evolution, components, main concepts (file, process, system calls). Basic hardware aspects: CPU, user and kernel mode, memory layers, I/O devices. Basic OS structure.	(1) use beamer slides, combined with blackboard illustration; (2) interactions with students: ask their opinion relative to the presented subject; (3) give each class a short evaluation test; let students discuss and argue	
2	**The Shell (Command Interpreter)**. Definition, role, functionality, simple and complex commands. Standard input and output redirection.		
3	**File systems (1)**. User Perspective. File and directory concept from the user point of view (definition, role, characteristics, operations).		
4	**File systems (2)**. Windows and Linux File Systems. Permission rights and system calls.		
5	**File systems (3)**. Implementation aspects. Implementation strategies overview, space management and related problems, hard and symbolic links.		
6	**Process management**. Process model: definition, role, characteristics. Linux and Windows process management system calls.		

7	**Thread management**. Thread model: user vs. kernel threads, implementation problems, usage, performance aspects. Basic scheduling algorithms (FIFO, SJF, Priority-based). Linux and Windows process thread system calls.	each other their solution; give them the good solution and let them evaluate their own one; (4) propose 2-3 interesting study cases of OSes to be prepared and presented by students; (5) students are invited to collaborate in research projects.	
8	**Process synchronization (1)**. Theoretical aspects. Context, definition, synchronization mechanisms, techniques and problems (locks, semaphores, monitors, mutual exclusion, starvation, deadlock).		
9	**Process synchronization (2)**. Classical synchronization patterns: producer/consumer, readers/writers, rendez-vous, barrier, dining philosopher, sleeping barber. Similarities between different synchronization mechanisms.		
10	**Inter-process communication**. Pipe files, shared memory, message queues, signals.		
11	**Memory management (1)**. Context, definition, binding, basic techniques, space management, addresses translation, swapping.		
12	**Memory management (2)**. Paging and segmentation.		
13	**I/O Devices Management**. Principles, disks, clocks, character-oriented terminals.		
14	**Security aspects**. Security policies and mechanisms. Basic program's vulnerabilities (buffer overflow).		

Bibliography

1. Andrew Tanenbaum. *Modern Operating System*, 2nd Edition, Prentice-Hall, 2005, ISBN 0-13-092641-8.
2. A. Silberschatz, P. Galvin, G. Gagne, Operating Systems Concepts, 8th Edition, Wiley, 2010
3. Remzi H. Arpaci-Dusseau, Andrea C. Arpaci-Dusseau, Operating Systems: Three Easy Pieces, online available at http://pages.cs.wisc.edu/~remzi/OSTEP/

8.2. Applications (Laboratory)		Teaching methods	Notes
1	**Laboratory presentation**: Purpose, contents, strategies, requirements. **Get familiar with Linux OS**: main characteristics, basic commands, access rights.	(1) students are presented a very brief overview of the most important and difficult aspects of the working subject; (2) students are given at the beginning of each class a short evaluation quiz; (3) students are given a hands-on tutorial to practice with working subject's aspects and to solve problems (4) students are given challenging problems for extra credit;	
2	**Linux batch scripts**: basic Linux commands, command line structure, scripts, command line parameters, variables, control flow commands, functions.		
3	**Linux system calls to access data in files**: basic system calls to store and retrieve data to and from regular user files: open, read, write, lseek, close.		
4	**Linux system calls for file and directory manipulation**: system calls to rename or remove a file, link a file to more directories, get information about a file or directory, change permission rights and listing a directory contents.		
5	**Windows case**: NTFS and FS system calls.		
6	**Linux system calls for process management**: system calls for creating a new process, terminating an existing process, waiting for a child process to terminate, loading another executable into an existing process etc.		
7	**Linux threads**: Linux implementation of POSIX functions used to create and manage threads: pthread_create, pthread_join, pthread_exit etc.		
8	**Synchronization mechanisms (1)**: Linux semaphores. Linux system calls to create and use semaphores: semget, semctl, semop.		
9	**Synchronization mechanisms (2)**: POSIX locks and condition variables. Linux functions used to create and use POSIX locks and condition variables: pthread_mutex_lock, pthread_mutex_unlock, pthread_cond_wait, pthread_cond_signal.		
10	**Windows Case**: process and thread system calls, synchronization mechanisms.		
11	**Inter-process Communication Mechanisms (IPC)**: Linux named (FIFO) and nameless pipes. System calls for managing and using pipes: pipe and mkfifo.		
12	**Memory management**: ELF executable file format. Virtual vs. physical address space. Dynamically allocated memory.		
13	**Memory management**: memory-mapped files, shared memory.		

14	**Security aspects**: buffer overflow detection and correction.		

Bibliography

1. Lecture slides and laboratory text and support at http://moodle.cs.utcluj.ro/
2. M. Mitchell, J. Oldham, A. Samuel, Advanced Linux Programming, New Riders Publishing, 2001

9. Bridging course contents with the expectations of the representatives of the community, professional associations and employers in the field

OS knowledge is a fundamental requirement in the CS field. We follow the ACM curricula guide. We also consult relevant IT companies about their practical expectations regarding OS knowledge and adapt accordingly our course contents. In this sense, Linux and Windows are the most used OSes. Usually the teachers in charge of lab classes are former graduate students of our CS program with consistent experience in industry. They are permanently consulted regarding the OS course curriculum and its applicability in real projects in industry.

10. Evaluation

Activity type	10.1 Assessment criteria	10.2 Assessment methods	10.3 Weight in the final grade
Course	Students must understand fundamental OS concepts and be able to correctly define them. They must also be able to apply their knowledge to solve user-space problems related to or dependent by an OS.	Small problem-like subjects requiring students to apply the theoretical learned OS related aspects to give a solution to proposed problem.	0.67
Applications	Students must be able to develop C programs that use different OS system calls to solve practical, problems related to or dependent by an OS.	Quiz tests. Programming problems, whose solution has to be implemented in C and run on computers.	0.33

10.4 Minimum standard of performance.

Students must attend minimum **9 lecture classes** to be allowed to take the exam in the regular exam session. Students must attend minimum **7 lecture classes** to be allowed to take the exam in any re-examination sessions. Less than 7 attended lecture classes leads to the interdiction to take any course re-examination in the university year the course is taught.

Students must attend minimum **12 lab classes** to be allowed to take the exam in the regular exam session. Students must attend minimum **10 lab classes** to be allowed to take the exam in any re-examination sessions. Less than 10 attended lab classes leads to the interdiction to take any lab re-examination in the university year the course is taught.

Students are allowed to take the final course examination only after passing the lab examination.

Be able to define the fundamental OS principles and concepts, like process, thread, file, directory, lock, semaphore, paging.

Be able to write C program to use fundamental system calls in Linux for working with files, processes, threads, synchronization mechanisms and memory.

Course responsible
Conf.dr.ing. Adrian Colesa

Head of department
Prof.dr.eng. Rodica Potolea

SYLLABUS

1. Data about the program of study

1.1	Institution	The Technical University of Cluj-Napoca
1.2	Faculty	Automation and Computer Science
1.3	Department	Computer Science
1.4	Field of study	Computer Science and Information Technology
1.5	Cycle of study	Bachelor of Science
1.6	Program of study/Qualification	Computer Science/ Engineer
1.7	Form of education	Full time
1.8	Subject code	28.

2. Data about the subject

2.1	Subject name					Elements of Computer Assisted Graphics					
2.2	Subject area					Computer Science and Information Technology					
2.3	Course responsible/lecturer					Prof.dr.eng. Gorgan Dorian – dorian.gorgan@cs.utcluj.ro					
2.4	Teachers in charge of applications					Lect.dr.eng. Bacu Victor, As.eng. Constantin Nandra, {victor.bacu, constantin.nandra}@cs.utcluj.ro					
2.5	Year of study	II	2.6	Semester	4	2.7	Assessment	exam	2.8	Subject category	DF/OB

3. Estimated total time

Sem.	Subject name	Lecture	Applications			Lecture	Applications			Individual study	TOTAL	Credit
		[hours / week.]				[hours / semester]						
			S	L	P		S	L	P			
4	**Elements of Computer Assisted Graphics**	**2**	**-**	**2**	**-**	**28**	**-**	**28**	**-**	**48**	**104**	**4**

3.1	Number of hours per week	4	3.2	of which, course	2	3.3	applications	2
3.4	Total hours in the teaching plan	56	3.5	of which, course	28	3.6	applications	28

Individual study	Hours
Manual, lecture material and notes, bibliography	20
Supplementary study in the library, online and in the field	6
Preparation for seminars/laboratory works, homework, reports, portfolios, essays	10
Tutoring	3
Exams and tests	9
Other activities	0

3.7	Total hours of individual study	48
3.8	Total hours per semester	104
3.9	Number of credit points	4

4. Pre-requisites (where appropriate)

4.1	Curriculum	Computer programming (C language)
4.2	Competence	Applications development in C programming language

5. Requirements (where appropriate)

5.1	For the course	Projector, computer
5.2	For the applications	Laboratory attendance is mandatory Study of laboratory materials from the server

6. Specific competences

Professional competences	**C3 –** Problems solving using specific Computer Science and Computer Engineering tools (4 credits) **C3.1 –** Identifying classes of problems and solving methods that are specific to computing systems **C3.2 –** Using interdisciplinary knowledge, solution patterns and tools, making experiments and interpreting their results **C3.3 –** Applying solution patterns using specific engineering tools and mehods **C3.4 –** Evaluating, comparatively and experimentally, the available alternative solutions for performance optimization **C3.5 –** Developing and implementing informatic solutions for concrete problems
Cross competences	N/A

7. Discipline objectives (as results from the *key competences gained*)

7.1	General objective	Learning about the architecture of a graphic system, the study of the graphic pipeline, the study of 2D graphic algorithms
7.2	Specific objectives	1. Creation of the graphical model of a scene of objects 2. Implementation of the basic algorithms that form the core of a graphic system 3. Development of graphic applications in a high-level programming language (C, C++) 4. Implementation of the main phases of the graphic transformation pipeline

8. Contents

8.1. Lecture (syllabus)		Teaching methods	Notes
1	Introduction. History. Examples	New multimedia teaching approaches will be used in classes. The course is interactive and includes demonstrations that exemplify graphical methods and algorithms.	During the semester and before each exam there are a few preparation hours planned.
2	Graphics systems – architecture, standards		
3	Graphics devices – logic and physics devices, input, output and interactive devices		
4	Graphics transformations pipeline – 2D and 3D transformations. Matrix operators		
5	Mathematics in computer graphics		
6	Lines scan conversion algorithms		
7	Circles scan conversion algorithms		
8	Polygons scan conversion algorithms		
9	Clipping algorithms – point, line, polygon and text		
10	Projections and viewing transformations		
11	Photorealistic presentation of 3D objects – concepts, algorithms, examples		
12	Color models – color perception, color space and standards, color in software design		
13	Graphics formats – vector and raster formats, data compression , Web technologies		
14	Graphics pattern grammars		

Bibliography

7. Foley J.D., van Dam, A., Feiner, S.K., Hughes, J.F., "*Computer Graphics. Principles and Practice*". Addison-Wesley Publishing Comp., 1992.
8. Watt A., "*3D Computer Graphics*". Addison-Wesley, 1998.

In virtual library

1. Course resources, http://cgis.utcluj.ro/teaching/

8.2. Applications (Laboratory)		Teaching methods	Notes
1	Introduction to SDL	Documentation and examples will be available to the students, prior to the laboratory	Each student will have to develop a specific
2	Mathematics in computer graphics: vectors		
3	Mathematics in computer graphics: matrices		
4	Graphics transformations		
5	Graphics transformations in SDL		

6	Line rasterization using the Bresenham algorithm	classes, on a dedicated server. The students will work independently but will also be assisted by the teacher.	project based on the knowledge acquired at the laboratory hours.
7	Clipping algorithms for graphical primitives		
8	Viewing transformations		
9	Triangle rasterization using barycentric coordinates		
10	Intermediate assessment		
11	Hidden surface removal using the z-buffer algorithm		
12	Bezier curves		
13	Color computation		
14	Final assessment		

Bibliography

In virtual library

1. Course and practical works, http://cgis.utcluj.ro/teaching/

9. Bridging course contents with the expectations of the representatives of the community, professional associations and employers in the field

This discipline is integrated into the Computers and Information Technology domain. The content is classic, yet modern, and introduces to students the fundamentals of graphic systems and 2D algorithms. The content of this discipline has been aligned with the information presented in similar disciplines from other major universities and companies from Romania, Europe and USA and has been evaluated by the authorized Romanian governmental agencies (CNEAA and ARACIS).

10. Evaluation

Activity type	10.1	Assessment criteria	10.2	Assessment methods	10.3	Weight in the final grade
Course		The written exam evaluates the understanding of the information presented in classes and the ability to apply this knowledge.		Evaluation is performed through written exam.		60% (E)
Course activity		The activity in class evaluates the active involvement of the students in the teaching process and their participation to the discussions, debates and other class activities during the entire semester.		Evaluation is performed through a very short tests.		10% (AC)
Applications		Laboratory assessment evaluates the practical abilities obtained by the students. Through homework assignments the students have the opportunity to develop their skill in applying the notions, concepts and methods presented in class.		Evaluation is performed through written and practical exam.		40% (L)
10.4 Minimum standard of performance						
Graduation requirement: M≥5; final mark M=0.5*E+0.4*L+0.1*AC						

Course responsible
Prof.dr.ing. Dorian Gorgan

Head of department
Prof.dr.eng. Rodica Potolea

SYLLABUS

1. Data about the program of study

1.1	Institution	The Technical University of Cluj-Napoca
1.2	Faculty	Automation and Computer Science
1.3	Department	Computer Science
1.4	Field of study	Computer Science and Information Technology
1.5	Cycle of study	Bachelor of Science
1.6	Program of study/Qualification	Computer Science/ Engineer
1.7	Form of education	Full time
1.8	Subject code	29.

2. Data about the subject

2.1	Subject name					Foreign Language II (English, French, German - Technical documents elaboration)					
2.2	Subject area					Computer Science and Information Technology					
2.3	Course responsible/lecturer					Lector dr. Sanda Paduretu					
2.4	Teachers in charge of applications					-					
2.5	Year of study	II	2.6	Semester	4	2.7	Assessment	Colloquium	2.8	Subject category	DC/OB

3. Estimated total time

Sem.	Subject name	Lecture	Applications			Lecture	Applications			Individual study	TOTAL	Credit
		[hours / week.]				[hours / semester]						
			S	L	P		S	L	P			
4	**Foreign Language II (English, French, German - Technical documents elaboration)**	**2**	**-**	**-**	**-**	**28**	**-**	**-**	**-**	**24**	**52**	**2**

3.1	Number of hours per week	2	3.2	of which, course	2	3.3	applications	-
3.4	Total hours in the teaching plan	28	3.5	of which, course	28	3.6	applications	-
Individual study								Hours
Manual, lecture material and notes, bibliography								
Supplementary study in the library, online and in the field								
Preparation for seminars/laboratory works, homework, reports, portfolios, essays								24
Tutoring								
Exams and tests								
Other activities								

3.7	Total hours of individual study	24
3.8	Total hours per semester	52
3.9	Number of credit points	2

4. Pre-requisites (where appropriate)

4.1	Curriculum	None
4.2	Competence	Minimum B2 level (CEFR)

5. Requirements (where appropriate)

5.1	For the course	N/A
5.2	For the applications	Class attendance, individual study

6. Specific competences

Professional competences	N/A
Cross competences	**CT3 –** Demonstrating the spirit of initiative and action for updating professional, economical and organizational culture knowledge (2 credits)

7. Discipline objectives (as results from the *key competences gained*)

7.1	General objective	Students should acquire knowledge and integrated skills to communicate in a foreign language in professional (technical and engineering) contexts and on job related topics.
7.2	Specific objectives	At the end of this course, the students will be able to: - identify and apply the main principles of effective communication in English - read and write using effective academic and technical writing techniques; -participate and express their opinion, evaluation and recommendation in technical exchange of information; -take notes on specialized topics within their field of specialization; -have the necessary skills read and write scientific articles -read and extract specific and general information from a variety of technical texts;

8. Contents

8.1. Lecture (syllabus)		Teaching methods	Notes
1	Introduction to communication. Communication in an academic setting. Communication at work.	Lecture by teacher, drill and practice, class discussion , questions and answers, textbook / reading assignments, formative assessment	
2	The writing process. Features and stages of the writing process.		
3	Readability. Characteristics and formulae for readability.		
4	Improving readability. Web-page / computer programming readability.		
5	Fundamentals of effective technical writing.		
6	Overview of technical and scientific language used in written communication. Best words and phrases. Reading grammar. Formal and informal language.		
7	Paragraphs. What is a paragraph? Elements of a paragraph. Development of a paragraph.		
8	Basic types of documents. User manuals, technical reports.		
9	Citation: plagiarism, paraphrasing, summary, academic conventions		
10	Plagiarism I: Complexities of definition. Plagiarism in Academic contexts. The Academy's response to plagiarism		
11	Plagiarism II: Learning to write from sources. The "shock" of referencing. Avoiding plagiarism.		
12	Plagiarism III: The art of finding plagiarism. Types of academic misconduct (ghost-writing, contract cheating, falsifying data).		
13	Plagiarism IV: Student's research on typologies of plagiarism. Assignment discussion. Identifying main types (copy-paste, verbatim, translations, disguised, shake and paste, clause quilts, structural, cut and slide, self-plagiarism).		
14	Style. Final conclusion.		
Bibliography			

1.Marinela Granescu, Ema Adam, Effective academic and technical writing, UTPress, Cluj-Napoca, 2010
2. Justine Jobel, Writing for Computer Science: the art of effective communication, Springer Verlag, Melbourne, 2000
3. Simon Haines, Real writing with answers, Cambridge University Press, 2008
4. R.R. Jordan, Academic writing course, Nelson, 1992

8.2. Applications (Seminars, Laboratory, Projects)		Teaching methods	Notes
1	-		

Bibliography
-

9. Bridging course contents with the expectations of the representatives of the community, professional associations and employers in the field

Mastering the elements of effective academic and technical writing will help the students in the field of computer science to integrate better in the labour market and improve personal development. The introduction in the language for specific purposes and academic discourse will facilitate reading and writing more documents in the field of study, making informed decisions on various types of information, and keeping up-to-date with state of the art knowledge in students' professional field. Most engineers or scientists work in organizational settings where team work is essential and good team work is impossible without good communication.

10. Evaluation

Activity type	10.1	Assessment criteria	10.2	Assessment methods	10.3	Weight in the final grade
Course		Completion of end-term evaluation, individual study, attendance to course		On-going class-work evaluation, and one end-term test (integrated skills)		Class-work evaluation - 20% End-term test – 80%
Applications						
10.4 Minimum standard of performance						
at least 50% of all components of tasks solved correctly						

Course responsible
Lect.dr. Sanda Paduretu

Head of department
Conf.univ.dr. Ruxanda Literat