SYLLABUS

1. Information on the study programme

1.1. Higher education institution	West University of Timișoara
1.2. Faculty	Matematics and Computer Science
1.3. Department	Computer Science (Informatică)
1.4. Study program field	Computer Science (Informatică)
1.5. Study cycle	Bachelor
1.6. Study programme / Qualification	Computer Science: Database administration / *Administrator baze de date - 252101; Computer network administration / Administrator de retea de calculatoare - 252301; Analyst / Analist - 251201; Research assistant in computer science / Asistent de cercetare în informatica - 214918; Teacher in secondary schools / Profesor în învatamântul gimnazial - 233002; Programmer / Programator - 251202; Software systems designers / Proiectant sisteme informatice - 251101*

2. Information on the course

2.1. Course title			Software Engineering				
2.2. Lecture instructor			Mîndruţă Cristina				
2.3. Seminar / laboratory instructor			Mîndruţă Cristina				
2.4. Study year	2	2.5. Semester	2	2.6. Examination type E(xam)/C(olloquim)	E	2.7. Course type M(andatory)/ E(lective)/ F(acultative)	M

3. Estimated study time (number of hours per semester)

3.1. Attendance hours per week	4	out of which: 3.2 lecture	2	3.3. seminar/ laborator	2
3.4. Attendance hours per semester	56	out of which: 3.5 lecture	28	3.6. seminar/ laborator	28

Distribution of the allocated amount of time	**hours/ ore**
Study of literature, course handbook and personal notes	30
Supplementary documentation at library or using electronic repositories	10
Preparing for laboratories, homework, reports etc.	30
Exams	5
Tutoring	5

3.7. Total number of hours of individual study	80
3.8. Total number of hours per semester	56
3.9. Number of credits (ECTS)	5

4. Prerequisites (if it is the case)

4.1. curriculum / de curriculum	Programming I, II, III

4.2. skills / de competențe	Programming în Java

5. Requirements (if it is the case)

5.1. for the lecture	Room with projector
5.2. for the seminar, laboratory	Room with personal computers; VP Community edition installed

6. Specific acquired competences

Professional competences	• Ability to understand and operate with fundamental concepts in software engineering • Ability to understand, analize and apply the software process
Transversal competences	• Ability of oral and written communication on professional topics with specialists and non-specialists in informatics and of writing technical reports and documentation in at least one international language • Ability to work individually and in a team and to obey specific ethical rules • Ability to learn new concepts and to quickly adapt to the new technologies which appear in informatics

7. Course objectives

7.1. General objective	• Provide students with concepts and main issues in software engineering and form abilities for professional and ethical approach of the development of software systems.
7.2. Specfic objectives	•*Ob. de cunoaștere (OC)*: (1) to explain the basic concepts of software engineering and the phases of the software development process, (2) to describe and compare different models of the software process. •*Ob. de abilitare (OAb)*: (1) to analyze user requests, to identify solutions, to compare and select software tools appropriate for solving a given problem. (2) to appropriately use basic UML diagrams (UC, activity, class, sequence, state transition) for the analysis and design of software systems. •*Ob. atitudinale (OAt)*: (1) to argue the importance of software engineering topics and of the principles of ethics of the software engineering profession. (2) to develop a proper relation with clients.

8. Content

8.1. Lecture	**Teaching methods**	**Remarks, details**
(2h) Introduction. Concepts and definitions. Professional and ethical responsibility. (OC1, OAt1)	Systematic explaining, examples, dialog	
(2h) Software requirements and requirements engineering process (OC1, Oab1, OAt2)	Systematic explaining, examples, dialog	
(2h) Software systems modelling (OC1, Oab1, OAb2)	Systematic explaining, examples, dialog	

(6h) Software systems design (OC1, OAb1, OAb2)	Systematic explaining, examples, dialog	
(2h) Software systems coding and debugging. Software systems evolution. (OC1, OAb1)	Systematic explaining, examples, dialog	
(2h) Software configuration management (OC1, OAb1)	Systematic explaining, examples, dialog	
(4h) Verification and validation. Software testing (OC1, OAb1, OAt2)	Systematic explaining, examples, dialog	
(4h) Software process and software process models (OC2, OAt2)	Systematic explaining, examples, dialog	
(2h) Software reuse (OC1)	Systematic explaining, examples, dialog	
(2h) Elements of software project management (OC1, OC2)	Systematic explaining, examples, dialog	

Recommended literature

1. Ian Sommerville, "Software Engineering" Eighth Edition, Addison-Wesley, 2007.
1. Steve McConnell, Code Complete, Second Edition, Microsoft Press, 2004
2. Scott W. Ambler, "The Elements of UML 2.0 Style", Cambridge University Press, 2005.
3. Tom Pender, "UML Bible", John Wiley & Sons, 2003

8.2. Seminar / laboratory	**Teaching methods**	**Remarks, details**
(2h) Software tools for UML. Individual study. (OAb1)	Discovery by documentation and examples study. Exercises and dialog. Individual practice.	
(2h) Functional modelling. Use cases. (OAb2)	Diagram description. Individual practice.	
(2h) Behavior modelling. Activity diagram (OAb2)	Diagram description. Individual practice.	
(2h) User interaction modelling. GUI prototyping. (OAb2, OAt2)	Method description. Individual practice.	
(2h) Quiz	Quiz and discussion	
(4h) Class diagram. Domain modelling. (OAb2)	Diagram and method description. Individual practice.	
(2h) Robustness analysis. Robustness diagram. (OAb2)	Method and diagram description. Individual practice.	
(2h) Sequence diagram (OAb2)	Diagram description. Individual practice.	
(2h) Quiz. Software processes. Example : ICONIX (OAb1)	Quiz and discussion. Individual study	
(2h) Coding good practices. (OAb1, OAb2)	Exemplification and individual study.	
(2h) Software verification. Test cases design. Unit testing and TDD. (OAb1)	Method description. Individual practice.	
(2h) Software validation (OAb1, OAt1)	Individual study. Dialog.	
(2h) Design patterns. Examples. (OAb1)	Design patterns. Examples.	

Recommended literature

1. http://www.agilemodeling.com
2. D. Rosenberg and M. Stephens, Use Case Driven Object Modeling with UMLTheory and Practice, Apress, 2007
3. http://www.visual-paradigm.com

4. http://www.cragsystems.co.uk
5. www.junit.org

9. Correlations between the content of the course and the requirements of the IT field and relevant employers

Software process knowledge and experience is necessary to any candidate for a job in a software development company. Client companies for software products can also benefit from the skills of a software engineer.

10. Evaluation

Activity	10.1. Assesment criteria	10.2. Assesment methods	10.3. Weight in the averaged mark / Pondere din nota finală
10.4. Lecture	Knowledge of fundamental theoretical elements presented and discussed in lectures	Written paper	50%
10.5. Seminar/ laboratory	Ability to follow a software development process. Ability to use the basic UML diagrams. Ability to define and aplly test cases.	Homework and quizes	50%
10.6. Minimal needed performance for passing			
Knowledge of fundamental theoretical elements. Good skills of modeling simple specifications in UML. Good skills in defining test cases. Using a software tool for UML modeling. Ability to undrerstand and follow a software development process.			

Date of completion 1.10.2016 | Signature (lecture instructor) | Signature (seminar instructor)

Date of approval | Signature (director of the department)
Conf.dr. Victoria Iordan