SYLLABUS

1. Information regarding the programme

1.1 Higher education institution	**Babes-Bolyai University**
1.2 Faculty	**Faculty of Mathematics and Computer Science**
1.3 Department	**Department of Computer Science**
1.4 Field of study	**Computer Science**
1.5 Study cycle	**Bachelor**
1.6 Study programme / Qualification	**Computer Science**

2. Information regarding the discipline

2.1 Name of the discipline (en) (ro)			**Software Systems Verification and Validation**				
2.2 Course coordinator			**PhD Lecturer Vescan Andreea**				
2.3 Seminar coordinator			**PhD Lecturer Vescan Andreea**				
2.4. Year of study	**3**	2.5 Semester	**6**	2.6. Type of evaluation	**E**	2.7 Type of discipline	**compulsory**
2.8 Code of the discipline		**MLE5040**					

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

3.1 Hours per week	4	Of which: 3.2 course	2	3.3 seminar/laboratory	2
3.4 Total hours in the curriculum	48	Of which: 3.5 course	24	3.6 seminar/laboratory	24
Time allotment:					hours
Learning using manual, course support, bibliography, course notes					22
Additional documentation (in libraries, on electronic platforms, field documentation)					22
Preparation for seminars/labs, homework, papers, portfolios and essays					22
Tutorship					3
Evaluations					8
Other activities:					0

3.7 Total individual study hours	77
3.8 Total hours per semester	125
3.9 Number of ECTS credits	5

4. Prerequisites (if necessary)

4.1. curriculum	•
4.2. competencies	•

5. Conditions (if necessary)

5.1. for the course	• Video projector, Internet access
5.2. for the seminar /lab activities	• Laboratory with computers; various tools for verification activities

6. Specific competencies acquired

Professional competencies	• Identification of proper methodologies for software systems development; Identification and explication of proper software systems specification methods; • Using methodologies and tools for development of informatics applications; Using proper criteria and methods for evaluation of software applications; • Realization of dedicated information projects.
Transversal competencies	• Application of efficient and rigorous working rules, manifest responsible attitudes toward the scientific and didactic fields, respecting the professional and ethical principles. • Use of efficient methods and techniques for learning, information, research and development of abilities for knowledge exploitation, for adapting to the needs of a dynamic society and for communication in Romanian as well as in a widely used foreign language

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

7.1 General objective of the discipline	• To gain knowledge of partial correct and total correct algorithms • To gain knowledge of designing correct algorithms and proving the correctness hand-in-hand; • To learn the methods of program verification and validation; • To become used with building correct programs from specification;; • To develop a modern programming style.
7.2 Specific objective of the discipline	• Students will know how and which are the steps of an inspection, either of the source code or specification of each stage of the development of the software system. • Students will know to create test cases from the specification and from source code, that will help them develop a better and robust software system. • Students will know how to use tools for the management of testing process. • Students will know how to design test cases using various criteria (black-box, white-box).

8. Content

8.1 Course	Teaching methods	Remarks
1. Verification and validation. Program inspection	Interactive exposure Explanation Conversation Didactical demonstration	
2. Program testing (1): the concept of program testing; unit testing: testing	Interactive exposure Explanation	

criteria – black box testing	Conversation Didactical demonstration	
3. Program testing (2): the concept of program testing; unit testing: testing criteria – white box testing (cont.)	Interactive exposure Explanation Conversation Didactical demonstration	
4. Program testing (3): Levels of testing (unit, integration, system, regression, acceptance)	Interactive exposure Explanation Conversation Didactical demonstration	
5. Symbolic execution	Interactive exposure Explanation Conversation Didactical demonstration	
6. Testing Web applications	Interactive exposure Explanation Conversation Didactical demonstration	
7. The theory of program correctness. The evolution of the concept of program correctness. • Floyd's method for prooving correctness. • Hoare's axiomatisation method	Interactive exposure Explanation Conversation Didactical demonstration	
8. Dijkstra: the weakest precondition. Stepwise refinement from specifications	Interactive exposure Explanation Conversation Didactical demonstration	
9. Program Quality	Interactive exposure Explanation Conversation Didactical demonstration	
10. Model checking	Interactive exposure Explanation Conversation Didactical demonstration	
11. SQA,CMM..	Interactive exposure Explanation Conversation Didactical demonstration	
12. SPI , Cleanroom	Interactive exposure Explanation Conversation Didactical demonstration	

Bibliography

Books

[Fre10] FRENTIU, M., Verificarea si validarea sistemelor soft, Presa Universitara Clujeana, 2010

[Pres10] R. S. Pressman, Software engineering: a practinioner's approach, seventh edition, Higher Education, 2010

[Crs09] L. Crispin, J. Grecory, Agile testing: a practical guide for testers and agile teams, Addison-Wesley, 2009

[You08] M. Pezzand, M. Young, Software Testing and Analysis: Process, Principles and Techniques, John

Wiley & Sons, 2008
[Nai08] K. Naik, P. Tripathy, Software testing and quality assurance. Theory and Practice, A John Wiley & Sons, Inc., 2008
[Kat08] J. P. Katoen, Principles of Model Checking, MIT Press, May 2008
[Pat05] R. Patton, Software Testing, Sams Publishing, 2005
[Mye04] Glenford J. Myers, The Art of Software Testing, John Wiley & Sons, Inc., 2004
[Brn02] I. Brnstein, Practical software testing, Springer, 2002
[Mor90] Morgan, C., Programing from Specifications, Prentice Hall, NewYork, 1990.
[Dro89] DROMEY G., Program Derivation. The Development of Programs From Specifications, Addison Wesley Publishing Company, 1989.

Articles
[Kin75] J. Darringer, J. King, Applications of symbolic execution to program testing, 1975
[Dij75] DIJKSTRA, E., Guarded commands, nondeterminacy and formal derivation of programs, CACM, 18(1975), 8, pg.453-457.
[Hoa69] HOARE, C.A.R., An axiomatic basis for computer programming, CACM, 12(1969), pg.576-580, 583.

Tutorials
During lectures/seminars/laboratories tutorials will be given for each assignment.

8.2 Seminar / laboratory	Teaching methods	Remarks
1. Seminar 1/Laboratory 1 Inspection Inspection tool Issue traker tool Test management tool (TestLink)	Presentation, Conversation, Problematizations, Discovery, Other methods – individual study, exercises	
2. Seminar 2/Laboratory 2 Test cases using Black-box Testing (BBT) Test management tool (TestLink) Continuous Integration tool (Jenkins)	Presentation, Conversation, Problematizations, Discovery, Other methods – individual study, exercises	
3. Seminar 3/Laboratory 3 Test cases using White-box Testing (WBT) Test management tool (TestLink) Continuous Integration tool (Jenkins)	Presentation, Conversation, Problematizations, Discovery, Other methods – individual study, exercises	
4. Seminar 4/Laboratory 4 Levels of testing Test management tool (TestLink) Continuous Integration tool (Jenkins) Laboratory 4. **Practical Test BBT + WBT.**	Presentation, Conversation, Problematizations, Discovery, Other methods – individual study, exercises	
5. Seminar 5/Laboratory 5 Web testing Web testing tool (e.g. Selenium Web Driver) Test management tool (TestLink) Continuous Integration tool (Jenkins)	Presentation, Conversation, Problematizations, Discovery, Other methods – individual study, exercises	
6. Seminar 6/Laboratory 6 Correctness. Static analysis ESCJava2, JML	Presentation, Conversation, Problematizations, Discovery, Other methods – individual study,	

	exercises	

Bibliography

See references from Lectures.
For each seminar, students must be prepared. Various articles/chapters from books are required to be read previous to each seminar.

9. Corroborating the content of the discipline with the expectations of the epistemic community, professional associations and representative employers within the field of the program

- Students will know how to use tools for test management
- Students will know how to apply testing methods for a software products.
- Students will learn various verification and validation methods of a software system, to design test cases using various criteria (black-box testing, white-box testing)

10. Evaluation

Type of activity	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Share in the grade (%)
10.4 Course	At the end of the semester a written examination will give a mark E.	Written examination	50%
	During lectures hours, two quizzes are given. The mark Q is given.	Two quizzes examination during lectures hours	10%
10.5 Seminar/lab activities	The activity at seminaries, consisting from participation in solving the exercises and discussions, will be appreciate by a mark S.	Seminar activity	10%
	The activity at laboratories, consisting from participation in solving the exercises and discussions, will be appreciate by a mark L.	Laboratory activity	20%
	During laboratory 4, a practical lab activity. A mark P is given.	Practical test laboratory activity	10%

Remark .

- Seminar/Laboratory assignments/Practical laboratory work may not be redone in the retake session.
- Written exams can be taken during the retake session.
- Students from Previous Years to 2016-2017
    - All the above rules apply to students from previous years.
    - Seminar/Laboratory assignments and practical laboratory activity must be redone during

didactic activity time (in the 12 weeks before normal session).

- Laboratory activity: each student will come with it own semi-group.
- Late delivery of assignments will be penilized with 1 point for each week. Maximum 4 weeks are allowed to deliver an assignment. After the deadline, the assignment will be graded with 1.
- At least grade 5 (from a scale of 1 to 10) at written exam. The final grade computed with the given formula must be at least 5 in order to pass the exam.
  Final grade=50%WrittenExam+10Quizes+10Seminar+20%Laboratory+10PracticalTest

10.6 Minimum performance standards

- Students will learn and apply testing methods for a software product.
- Students will apply various methods for verification (testing, inspection, model checking) for establishing the correctness of an algorithm.
- Attend 75% of seminar activities during semester AND attend 90% of lab activities during semester.

Date	Signature of course coordinator	Signature of seminar coordinator
26 April 2016	Lect. PhD. Andreea Vescan,	Lect. PhD. Andreea Vescan

Date of approval	Signature of the head of department
.....	Prof. PhD. Anca Andreica