SYLLABUS

Software Systems Verification and Validation

University year 2025-2026

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	Computer Science
1.4. Field of study	Computer Science
1.5. Study cycle	Bachelor
1.6. Study programme/Qualification	Computer Science
1.7. Form of education	Full time

2. Information regarding the discipline

2.1. Name of the discipline	**Software Systems Verification and Validation**					Discipline code	**MLE5014**
2.2. Course coordinator					PhD Associate Professor Vescan Andreea		
2.3. Seminar coordinator					PhD Associate Professor Vescan Andreea		
2.4. Year of study	3	2.5. Semester	6	2.6. Type of evaluation	E	2.7. Discipline regime	Compulsory

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/project	**2**
3.4. Total hours in the curriculum	48	of which: 3.5 course	24	3.6 seminar/laboratory/project	**24**
Time allotment for individual study (ID) and self-study activities (SA)					**hours**
Learning using manual, course support, bibliography, course notes (SA)					33
Additional documentation (in libraries, on electronic platforms, field documentation)					33
Preparation for seminars/labs, homework, papers, portfolios and essays					34
Tutorship					7
Evaluations					20
Other activities:					0
3.7. Total individual study hours	**127**				
3.8. Total hours per semester	**175**				
3.9. Number of ECTS credits	**7**				

4. Prerequisites (if necessary)

4.1. curriculum	• Object oriented programming, Advanced programming methods, Systems for design and implementation, Web Programming
4.2. competencies	• Skills in highlevel object oriented programming environments

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

6.1. Specific competencies acquired [1]

[1] One can choose either competences or learning outcomes, or both. If only one option is chosen, the row related to the other option will be deleted, and the kept one will be numbered 6.

Professional/essential competencies	• advanced programming skills in high-level programming languages • development and maintenance of software systems • use of software tools in an interdisciplinary context
Transversal competencies	• efficient development of organized activities in an interdisciplinary group and the development of empathetic abilities for interpersonal communications, to relate to and cooperate with various groups • efficient development of organized activities in an interdisciplinary group and the development of empathetic abilities for interpersonal communications, to relate to and cooperate with various groups

6.2. Learning outcomes

Knowledge	The student knows: • The graduate has adequate knowledge related to the use of integrated development environments for creating large complex applications. • • The graduate is familiar with traditional and agile development methodologies.
Skills	The student is able to: • The graduate has the ability to create automated tests of different levels of granularity for quality assurance of the developed systems. • The graduate is familiar with tools used for testing, debugging, validating software applications. • The graduate is familiar with methods for testing and verifying software systems. • The graduate is familiar with project management tools, version control systems, and continuous integration/continuous delivery (CI/CD) concepts, methods, tools.
Responsibility and autonomy:	The student has the ability to work independently to obtain: • The graduate has the ability to understand and communicate information effectively. • The graduate has the ability to observe and obtain information from various sources.

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	
Program testing (1): the concept of program testing; unit testing: testing criteria – black box testing,	Interactive exposure Explanation Conversation Didactical demonstration	
Program testing (2): the concept of program testing; unit testing: testing criteria – white box testing (cont.)	Interactive exposure Explanation Conversation Didactical demonstration	
Program testing (3): Levels of testing (unit, integration, system, regression, acceptance)	Interactive exposure Explanation Conversation Didactical demonstration	
Testing Web applications	Interactive exposure Explanation Conversation Didactical demonstration	
Agile testing. Script testing versus exploratory testing	Interactive exposure Explanation Conversation Didactical demonstration	
Symbolic execution	Interactive exposure Explanation Conversation Didactical demonstration	
Model checking	Interactive exposure Explanation Conversation Didactical demonstration	
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	

Dijkstra: the weakest precondition.Stepwise refinement from specifications		
Program Quality	Interactive exposure Explanation Conversation Didactical demonstration	
Verification/testing related activities: Technical testing skills, Soft testing skills, Giving, feedback. This activity is done in collaboration of the teacher with the students.	Interactive exposure Explanation Conversation Didactical demonstration	
Final exam preparation.	Interactive exposure Explanation Conversation Didactical demonstration	

Bibliography

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
Seminar 1/Laboratory 1 Inspection Inspection tool	Presentation, Conversation, Problematizations, Discovery, Other methods – individual	

	study, exercises	
Seminar 2/Laboratory 2 Test cases using Black-box Testing (BBT) Continuous Integration tool (Jenkins)	Presentation, Conversation, Problematizations, Discovery, Other methods – individual study, exercises	
Seminar 3/Laboratory 3 Test cases using White-box Testing (WBT) Continuous Integration tool (Jenkins)	Presentation, Conversation, Problematizations, Discovery, Other methods – individual study, exercises	
Seminar 4/Laboratory 4 Levels of testing - Integration testing Continuous Integration tool (Jenkins)	Presentation, Conversation, Problematizations, Discovery, Other methods – individual study, exercises	
Seminar 5/Laboratory 5 Web testing Web testing tool (e.g. Selenium Web Driver) Continuous Integration tool (Jenkins)	Presentation, Conversation, Problematizations, Discovery, Other methods – individual study, exercises	
Seminar 6/Laboratory 6 Correctness. Static analysis ESCJava2, JML	Presentation, Conversation, Problematizations, Discovery, Other methods – individual study, exercises	
Bibliography See references from Lectures. **Remark.** 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 product.
- 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

Activity type	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Percentage of final grade
10.1 Course	At the end of the semester a written examination will give a mark E.	Written examination	50%
10.2 Seminar/laboratory	The activity at seminaries, consisting from participation in solving the exercises and discussions will	Seminar = Grade for seminar Activity	25%

	be appreciate by a mark S.		
	The activity at laboratories, consisting from participation in solving the exercises and discussions, will be appreciate by a mark L.	Laboratory activity	25%
10.3 Bonus point	Students will have the possibility of obtaining bonus points at the final grade for additional activities that are related to Software systems verification and validation: conduction research/report and various activities during lectures. An R&D project could also be selected.	Bonus points	Bonus points at the final grade (after obtaining the final minimum required grade 5).

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 the current academic year
    - 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.
- Laboratory activity: 3 out of 6 laboratories must be delivered.
- Late delivery of assignments will be penilized. Maximum 4 weeks are allowed to deliver an assignment. After the deadline, the assignment will be graded with 0.
- The final grade computed with the given formula must be at least 5 in order to pass the exam.
  Final grade=50%WrittenExam+25%Seminar+25%Laboratory

Attend 75% of seminar activities during semester AND attend 90% of lab activities during semester.

10.6 Minimum standard of performance
•

11. Labels ODD (Sustainable Development Goals)[2]

[2] Keep only the labels that, according to the *Procedure for applying ODD labels in the academic process*, suit the discipline and delete the others, including the general one for *Sustainable Development* – if not applicable. If no label describes the discipline, delete them all and write „*Not applicable.*".

Not applicable.

Date: ...	Signature of course coordinator	Signature of seminar coordinator ....

Date of approval: ...	Signature of the head of department Assoc.prof.phd. Adrian STERCA