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	**Mathematics**
1.5 Study cycle	**Bachelor**
1.6 Study programme / Qualification	**Mathematics Computer Science (in Romanian)**

2. Information regarding the discipline

2.1 Name of the discipline	**Test Design Techniques (Tehnici de testare software)**						
2.2 Course coordinator	**Lecturer PhD Camelia Chisăliţă-Creţu**						
2.3 Seminar coordinator	**Lecturer PhD Camelia Chisăliţă-Creţu**						
2.4. Year of study	**3**	2.5 Semester	**6**	2.6. Type of evaluation	**E**	2.7 Type of discipline	**Optional**
2.8 Discipline Code	**MLE5110**						

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

3.1 Hours per week	5	Of which: 3.2 course	2	3.3 seminar/laboratory	1 lab + 1 sem + 1 project
3.4 Total hours in the curriculum	60	Of which: 3.5 course	24	3.6 seminar/laboratory	36

Time allotment:	Hours
Learning using manual, course support, bibliography, course notes	20
Additional documentation (in libraries, on electronic platforms, field documentation)	20
Preparation for seminars/labs, homework, papers, portfolios and essays	40
Tutorship	6
Evaluations	4
Other activities:	-

3.7 Total individual study hours	90
3.8 Total hours per semester	150
3.9 Number of ECTS credits	6

4. Prerequisites (if necessary)

4.1. curriculum	• OOP, Programming Fundamentals, Advanced Programming Methods
4.2. competencies	• Good programming skills in at least one of the programming languages Java, C#

5. Conditions (if necessary)

5.1. for the course	• Course hall with projector
5.2. for the seminar /lab activities	• Laboratory: computers and use of a programming language

	environment

6. Specific competencies acquired

Professional competencies	• **C2.1** Identify adequate software systems development methodologies • **C1.2** Identify and explain specific test design techniques that correspond to a testing level. • **C1.3** Source code and goal oriented test elaboration in a well-known programming language. • **C4.3** Identify models and methods adequate to real life problem solving.
Transversal competencies	• **CT1** Apply rules to organized and efficient work, responsibilities of didactical and scientific activities and creative capitalization of own potential, while respecting principles and rules for professional ethics. • **CT3** Use efficient methods and techniques for learning, knowledge gaining, and research and develop capabilities for capitalization of knowledge, accommodation to society requirements and communication in English.

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

7.1 General objective of the discipline	• Enhance the students understanding of testing and test design techniques. • Provide the students with an environment in which they can explore the usage and usefulness of software testing and test design concepts in various business scenarios. • Induce a realistic and industry driven view of software testing concepts and their inherent benefits.
7.2 Specific objective of the discipline	• Give students the ability to explore various test design techniques applied to different levels of testing. • Improve the students' abilities to tackle on goal driven testing. • Enhance the students understanding of test design techniques value in business. • Students will be able to use various tools for the testing process (i.e., test management, test running, test reporting and bug reporting). • Students will be able to design test cases according to an established testing goal and using specific test design technique in order to investigate the software.

8. Content

8.1 Course	Teaching methods	Remarks
1. Software Testing. Test Design Techniques 1.1. Software Testing. Goals. Scope 1.2. Test Design Technique. Attributes 1.3. Taxonomy of Test Design Techniques	• Interactive exposure • Explanation. Conversation • Didactical demonstration	
2. Coverage-based Techniques I 2.1. Focus. Objectives 2.2. Tours. Logical Expressions	• Interactive exposure • Explanation. Conversation • Didactical demonstration	
3. Coverage-based Techniques II 3.1. Specification-based Testing; 3.2. Requirements-based Testing;	• Interactive exposure • Explanation. Conversation • Didactical	

	demonstration	
4. Risk-based Techniques 4.1. Focus. Objectives 4.2. Quick-tests. History-based Testing. Usability Testing 4.3. HTSM. Failure modes	• Interactive exposure • Explanation. Conversation • Didactical demonstration	
5. Activity-based Techniques 5.1. Focus. Objectives 5.2. Guerilla Testing. All-pairs Testing 5.3. Use Cases Testing. Scenario Testing Coverage-based Techniques vs Activity-based Techniques	• Interactive exposure • Explanation. Conversation • Didactical demonstration	
6. Evaluation-based Techniques 6.1. Focus. Objectives 6.2. Function Equivalence Testing. Self-verifying data	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
7. Desired result-based Techniques 7.1. Focus. Objectives 7.2. Confirmation Testing. User Acceptance Testing 7.3. Desired-based Techniques vs Evaluation-based Techniques	• Interactive exposure • Explanation. Conversation • Didactical demonstration	
8. Tester-based Techniques 8.1. Focus. Objectives 8.2. User Testing. Alpha Testing. Beta Testing 8.3. Bug Bashes. Paired Testing. 8.4. Coverage-based Techniques vs Tester-based Techniques	• Interactive exposure • Explanation. Conversation • Didactical demonstration	
9. Test Design Techniques Analysis 9.1. Tester-based Techniques vs Activity-based Techniques 9.2. Risk-based Techniques vs Coverage-based Techniques 9.3. Desired result-based Techniques vs Risk-based Techniques	• Interactive exposure • Explanation. Conversation • Didactical demonstration	
10. Security Testing 10.1.Terminology 10.2.Types 10.3.Advanced Techniques	• Interactive exposure • Explanation. Conversation • Didactical demonstration	
11. Bug Reporting 11.1.Challenges 11.2.RIMGEA Strategy	• Interactive exposure • Explanation. Conversation • Didactical demonstration	
12. Project Preparation	• Interactive exposure • Explanation. Conversation • Didactical demonstration	

Bibliography

[Kaner99] C. Kaner, J. Falk, H.Q. Nguyen, *Testing Computer Software*, Wiley, 1999.
[Brn02] I. Burnstein, *Practical Software Testing*, Springer, 2002.
[Kaner02] C. Kaner, J. Bach, B. Pettichord, *Lesson Learned in Software Testing*, Wiley, 2002.
[Mye04] Glenford J. Myers, *The Art of Software Testing*, John Wiley & Sons, Inc., 2004.

[Nai08] K. Naik, P. Tripathy, *Software testing and quality assurance. Theory and Practice*, A John Wiley & Sons, Inc., 2008.
[Crs09] L. Crispin, J. Grecory, *Agile testing: a practical guide for testers and agile teams*, Addison-Wesley, 2009.
[Pres10] R. S. Pressman, *Software engineering: a practinioner's approach*, seventh edition, Higher Education, 2010.
[BBST2008] BBST – *Bug Advocacy*, http://www.testingeducation.org/BBST/bugadvocacy/BugAdvocacy2008.pdf
[BBST2010] BBST – *Fundamentals of Testing*, Cem Kaner, http://www.testingeducation.org/BBST/foundations/BBSTFoundationsNov2010.pdf.
[BBST2011] BBST – *Test Design*, Cem Kaner, http://www.testingeducation.org/BBST/testdesign/BBSTTestDesign2011pfinal.pdf
[Whitt2012] J. Whittaker, J. Arbon J. Carollo, *How Google Tests Software*, Google, Pearson Education, 2012.
[OWASP2014] QWASP, *Testing guide 4.0*, 2014, https://owasp.org/www-project-web-security-testing-guide/assets/archive/OWASP_Testing_Guide_v4.pdf
[NRVR2014] Ana Filipa Nogueira, José Carlos Ribeiro, Francisco Fernández de Vega, Mário Alberto Zenha-Rela, *Object-Oriented Evolutionary Testing: A Review of Evolutionary Approaches to the Generation of Test Data for Object-Oriented Software*, International Journal of Natural Computing Research 4(4):15-35, October, 2014.
[KMS2014] Kaur, Manpreet and Rupinder Singh. *A Review of Software Testing Techniques*, 2014.
[Meer2014] Joris Meerts, *Functional Testing Heuritics*, https://www.testingreferences.com/docs/Functional_Testing_Heuristics.pdf
[Draghia2019] Claudiu Draghia, *Gamificarea in software testing. Testing Challenges*, http://testingchallenges.thetestingmap.org/, 2019.
[ForK2019] István Forgács, Attila Kovács, *Practical Test Design Selection of traditional and automated test design techniques*, BCS, 2019.
[BSR2021] F. A. Bhuiyan, M. B. Sharif and A. Rahman, *Security Bug Report Usage for Software Vulnerability Research: A Systematic Mapping Study*, IEEE Access, vol. 9, pp. 28471-28495, 2021, doi: 10.1109/ACCESS.2021.3058067.
[AIW2021] Samah W.G. AbuSalim, Rosziati Ibrahim, Jahari Abdul Wahab, *Comparative Analysis of Software Testing Techniques for Mobile Applications*, Journal of Physics: Conference Series, vol 1793, 2021.
[PLGM2022] Sheena Panthaplackel, Junyi Jessy Li, Milos Gligoric, Raymond J. Mooney, *Learning to Describe Solutions for Bug Reports Based on Developer Discussions*, ACL 2022, pp. 2935 – 2952.

8.2 Seminar	Teaching methods	Remarks
1. Seminar 1 Software Testing Terminology. Famous Bugs Posters	Presentation, Conversation, Problematizations, Discovery, Other methods – individual study, exercises	
2. Seminar 2 Black-Box Techniques. Coverage-based Techniques	Presentation, Conversation, Problematizations, Discovery, Other methods – individual study, exercises	
3. Seminar 3 Risk-based Techniques	Presentation, Conversation, Problematizations, Discovery, Other methods – individual study, exercises	
4. Seminar 4 Test Automation Tools. Demo	Presentation, Conversation, Problematizations, Discovery, Other methods – individual study, exercises	
5. Seminar 5 Activity-based Techniques. Desired result-based Techniques. White-Box Techniques. Coverage Criteria	Presentation, Conversation, Problematizations, Discovery, Other methods – individual study, exercises	
6. Seminar 6 Bug reporting. Bug stories. Games	Presentation, Conversation, Problematizations, Discovery, Other methods – individual study, exercises	

(puzzles, mnemonics)		
References: See references from Lectures.		
8.3 Laboratory	Teaching methods	Remarks
1. Laboratory 1 Software Testing Terminology. Testing Project Setup	Presentation, Conversation, Problematizations, Discovery, Other methods – individual study, exercises	
2. Laboratory 2 Coverage-based Techniques	Presentation, Conversation, Problematizations, Discovery, Other methods – individual study, exercises	
3. Laboratory 3 Risk-based Techniques	Presentation, Conversation, Problematizations, Discovery, Other methods – individual study, exercises	
4. Laboratory 4 Test Automation Tools	Presentation, Conversation, Problematizations, Discovery, Other methods – individual study, exercises	
5. Laboratory 5 Activity-based Techniques OR Desired result-based Techniques	Presentation, Conversation, Problematizations, Discovery, Other methods – individual study, exercises	
6. Laboratory 6 Bug reporting	Presentation, Conversation, Problematizations, Discovery, Other methods – individual study, exercises	
References: See references from Lectures.		

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 apply test design techniques for a software product, in a similar way they are used in industry.
- Students will be able to understand the differences between the goals and scope of the various test techniques applied to a software system.

10. Evaluation

Type of activity	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Share in the grade (%)
10.4 Course	Design and develop a testing solution (project) for a software product with focus on test design techniques. The corresponding grade is denoted by **P.**	Oral Examination	70%
10.5 Seminar/laboratory activities	Each lab activity will be graded. The arithmetic average of the grades is denoted by **L**.	Laboratory Activity	30%
Remark: • Laboratory assignments will pe achieved in groups of 2-3 students. • Testing project will pe achieved in groups of 4-5 students.			
10.6 Minimum performance standards			
• Students will be able to apply test design techniques according to established goals for a software system. • Students will be able to unstandand the differences between software testing goal, scope, and test design technique concepts.			

- The final grade (M) is computed as follows: M = 30%L+70%P.
- At least M >= 5.00 is favourable to pass this course exam.

Date	Signature of course coordinator	Signature of seminar coordinator
13.04.2023	**Lect. PhD. Camelia Chisăliţă-Creţu,**	**Lect. PhD. Camelia Chisăliţă-Creţu,**

Date of approval

Signature of the head of department

Prof. PhD. Laura Dioșan