



DISCIPLINE ACADEMIC SHEET
ACADEMIC YEAR 2025 – 2026

1. PROGRAMME DATA

1.1. Higher Education Institution	UNIVERSITY OF CRAIOVA
1.2. Faculty	Automation, Computers and Electronics
1.3. Department	Computers and Information Technology
1.4. Field of Study	Computers and Information Technology
1.5. Study level	L(License / Undergraduate)
1.6. Organizational form	IF
1.7. Study Program (name/code) /Qualification	Computers / L2060101010

2. Discipline data

2.1 Discipline Name	Software Engineering						
2.2 Course Activities Holder	Mihai-Cătălin SBORA						
2.3. Practical Activities Holder	Mihai-Cătălin SBORA						
2.4. Study Year	3	2.5. Semester	2	2.6. Evaluation type	E	2.7. Discipline regime	DOB

3. Estimated total time (hours per semester of teaching activities)

3.1. Number of hours per week	5	in which: 3.2 course	3	3.3. sem./lab./pr.	2
3.4. Total hours of curriculum	70	in which: 3.5 course	42	3.6. sem./lab./pr.	28
Time distribution - hours/week					
Study after manual, course support, bibliography and notes					14
Additional documentation in library, on specialized electronic platforms and field					5
Training seminars / labs, homework, portfolios and essays					5
Tutorial					
Examinations					2
Other activities: consultations					4
3.7. Total hours per individual activities					30
3.8. Total hours per semester					100
3.9. Number of credits					4

4. Preconditions (where appropriate)

4.1. of curriculum	● Object-Oriented Programming
4.2. of competence	● N/A

5. Conditions (where appropriate)

5.1. of the course	Teaching activities are carried out using a video projector. For situations when activities must be carried online, Google Classroom with Google Meet will be used. It provides access to course materials (lecture notes in
--------------------	--

	electronic format and other course documentation). The teaching process has the following structure: • 70% activity of theoretical presentation using lecture slides • 30% interactive activity (talks with students)
5.2. of seminar/ laboratory/project	Laboratory activities require Internet access. For situations when activities must be carried online, Google Classroom with Google Meet will be used. On each terminal used for laboratory activities several frameworks are installed, which cover mainly the design and implementation phases from the life cycle of a software application (e.g. .NET Core framework, free UML tools). Also, the students will use online services like GitHub, BitBucket, GitLab for tracking code changes and Diagrams.net for software design and architecture. Performing laboratory activities uses tutorials and practical demonstrations. Students have access to updated laboratory material. In addition, it is offered feedback for each software process from an application development.

6. COURSE OBJECTIVES – expected learning outcomes that are supported by completing and passing the course

Knowledge	Student/Graduate: 1. Understand the principles, processes, and methodologies that define software engineering.
Skills (Abilities)	Student/Graduate: 1. Analyze user and system requirements to produce clear, structured, and verifiable specifications. 2. Design and document robust and maintainable software architectures. 3. Apply testing and validation techniques to ensure software quality. 4. Understand and apply team-based development and project management methods. 5. Use modern tools and environments for software engineering (IDE, Git, CI/CD, issue tracking).
Responsibility and autonomy	Student/Graduate: 1. Assume responsibility for the design, implementation, and validation of software systems that meet specified functional and quality requirements. 2. Work autonomously in performing complex software engineering tasks, including coding, testing, documentation, and maintenance. 3. Collaborate effectively, taking responsibility for individual and collective contributions to project outcomes. 4. Adapt independently to new tools, frameworks, and methodologies in software development and project management. 5. Engage in continuous self-learning to maintain professional competence in a rapidly evolving technological environment.

7. Content

7.1. Course	The method of implementation	Teaching methods	Allocated time (hours)
1. Introduction to Software Engineering. Software Products	face to face	Teaching activities are carried out using a video projector or in online sessions when physical presence is not possible :	3
2. Agile Software Engineering	face to face		3

3.	Agile Requirements Engineering. Features, Scenarios and Stories	face to face	70% theoretical presentation using lecture slides 30% interactive activity (talks with students) Students have access to course materials in electronic format.	3
4.	Software Architecture	face to face		3
5.	Code Management and DevOps Automation	face to face		6
6.	Cloud-Based Software	face to face		3
7.	Microservices Architecture	face to face		6
8.	Security and Privacy	face to face		6
9.	Reliable programming	face to face		6
10.	Software testing	face to face		3
Bibliography:				
1. Engineering Software Products: An Introduction to Modern Software Engineering, Global Edition 1st Edition, Pearson, 2020				
2. Software Engineering, Ian Sommerville, Addison Wesley, 10th Edition, 2015				
3. Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise, Scott W. Ambler, Mark Lines, IBM Press, 2012				
4. Software Engineering: an engineering approach, James F. Peters and Witold Pedrycz, John-Wiely, 2000				
5. Design Patterns. E. Gamma. R. Helm. R. Johnson. J. Vlissides, Addison Wesley, 1995				

7.2. Seminar/laboratory/project	The method of implementation	Teaching methods	Allocated time (hours)
1. Agile software development	face to face	Laboratory sessions are made using tutorials and practical demonstrations. Emphasis is placed on understanding how to use the main techniques for object-oriented software development, and discussions with students, giving feedback for each problem solved	2
2. Requirements analysis with use cases and scenarios	face to face		2
3. Version control systems (Subversion, Git)	face to face		2
4. Continuous Integration/Continuous Delivery Pipelines	face to face		4
5. Design patterns	face to face		2
6. Software implementation (coding standards and common architectures, security)	face to face		8
7. Software testing (UnitTesting - MSTest)	face to face		4
8. Automated System Testing	face to face		4
Bibliography:			
1. Software Engineering, Ian Sommerville, Addison Wesley, 10th Edition, 2015 2. Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise, Scott W. Ambler, Mark Lines, IBM Press, 2012 3. Software Engineering: a practitioner's approach, Roger Pressman, Addison Wesley, 5th Edition, 2001 4. Design Patterns, E. Gamma, R. Helm, R. Johnson, J. Vlissides, Addison Wesley, 1995			

8. Correlating the content of the discipline with the expectations of representatives from the epistemic community, professional associations and representative employers in the field related to the program

The course content and its applications are consistent with similar disciplines from other universities in the country and abroad. Also, all the components are in line with the requirements of employers' representatives in the IT field. The outcomes of the course are discussed with representatives of local IT companies

9. Evaluation

Activity Type	9.1. Evaluation criteria	9.2. Evaluation methods	9.3. Final mark weight
9.4. Course	Understanding the methods and techniques for Agile requirement analysis - Understanding the methods and techniques used in an Agile Process - Understanding the methods and techniques for testing and verification - Understanding the methods for project planning - The ability to make connections between the concepts taught	Exam	60%
9.5. Seminar/laboratory/project	- The ability to apply theoretical knowledge in practice - Ability to develop software programs - Ability to present the results in a clear and concise manner	Continuous evaluation during the semester through individual/team assignments	40%
9.6. Minimum standard of performance			
- Using the basic knowledge for software requirements specification. - Using the basic knowledge for development operations (DevOps) - Using the basic knowledge for implementing cloud based and distributed software programs. - Using the basic knowledge for software testing			

Date of Completion	Signature of the Course Activities Holder,	Signature of the Practical Activities Holder,
01.10.2025	S.I dr. ing. Mihai-Cătălin SBORA	

Date of Approval in the Department	Signature of the Department Director,
03.10.2025	Ș.I.dr.ing. Nicolae ENESCU

Date of Approval in the Faculty	Signature of the Dean,
	Prof.dr.ing. Dorin ȘENDRESCU