# SYLLABUS

## 1. Data about the program of study

| | |
|---|---|
| 1.1 Institution | The Technical University of Cluj-Napoca |
| 1.2 Faculty | Faculty of Automation and Computer Science |
| 1.3 Department | Computer Science |
| 1.4 Field of study | Computer Science and Information Technology |
| 1.5 Cycle of study | Bachelor of Science |
| 1.6 Program of study/Qualification | Computer science/ Engineer |
| 1.7 Form of education | Full time |
| 1.8 Subject code | 33. |

## 2. Data about the subject

| | | | | | |
|---|---|---|---|---|---|
| 2.1 Subject name | *Software engineering* | | | | |
| 2.2 Course responsible/lecturer | Prof. dr eng. Eneia Todoran – Eneia.Todoran@cs.utcluj.ro | | | | |
| 2.3 Teachers in charge of seminars/ laboratory/ project | Assoc.prof. dr. Mitrea Paulina – Paulina.Mitrea@cs.utcluj.ro, Assoc.prof. dr. eng. Mitrea Delia Delia.Mitrea@cs.utcluj.ro | | | | |
| 2.4 Year of study | III | 2.5 Semester | 1 | 2.6 Type of assessment (E - exam, C - colloquium, V - verification) | E |
| 2.7 Subject category | DF – fundamentală, DD – în domeniu, DS – de specialitate, DC – complementară | | | | DD |
| | DI – Impusă, DOp – opțională, DFac – facultativă | | | | DI |

## 3. Estimated total time

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 3.1 Number of hours per week | 4 | of which: | Course | 2 | Seminars | Laboratory | 1 | Project | 1 |
| 3.2 Number of hours per semester | 56 | of which: | Course | 28 | Seminars | Laboratory | 14 | Project | 14 |
| 3.3 Individual study: | | | | | | | | |
| (a) Manual, lecture material and notes, bibliography | | | | | | | | 20 |
| (b) Supplementary study in the library, online and in the field | | | | | | | | 17 |
| (c) Preparation for seminars/laboratory works, homework, reports, portfolios, essays | | | | | | | | 17 |
| (d) Tutoring | | | | | | | | 5 |
| (e) Exams and tests | | | | | | | | 10 |
| (f) Other activities: | | | | | | | | 0 |

| | |
|---|---|
| 3.4 Total hours of individual study (suma (3.3(a)…3.3(f))) | 69 |
| 3.5 Total hours per semester (3.2+3.4) | 125 |
| 3.6 Number of credit points | 5 |

## 4. Pre-requisites (where appropriate)

| | |
|---|---|
| 4.1 Curriculum | Object Oriented Programming, Programming Techniques |
| 4.2  Competence | Competences acquired in the above disciplines |

## 5. Requirements (where appropriate)

| | |
|---|---|
| 5.1. For the course | Blackboard / whiteboard, internet, projector, computer |
| 5.2. For the applications | Computers, internet, specific software |

## 6. Specific competence

| | |
|---|---|
| 6.1 Professional competences | **C3 -** Problems solving using specific Computer Science and Computer Engineering tools (2 credits) |
| | **C3.1 -** Identifying classes of problems and solving methods that are specific to computing systems |
| | **C3.2 -** Using interdisciplinary knowledge, solution patterns and tools, making experiments and interpreting their results |
| | **C3.3 -** Applying solution patterns using specific engineering tools and mehods |
| | **C3.4 -** Comparatively and experimentaly evaluation of the alternative solutions |

| | for performance optimization<br>**C3.5** - Developing and implementing informatic solutions for concrete problems<br>**C4 -** Improving the performances of the hardware, software and communication systems (1 credit)<br>**C4.1 -** Identifying and describing the defining performance elements of hardware, software and communication systems<br>**C4.2 -** Explaining the interaction of the factors that determine the performances of hardware, software and communication systems<br>**C4.3 -** Applying fundamental methods and principles for increasing performance of hardware, software and communication systems<br>**C4.4 -** Choosing criteria and methods for performance evaluation of hardware, software and communication systems<br>**C4.5 -** Developing performance based professional solutions for hardware, software and communication systems<br>**C5 -** Designing, managing the lifetime cycle, integrating and ensuring the integrity of hardware, software and communication systems (2 credits)<br>**C5.1 -** Specifying the relevant criteria regarding the lifetime cycle, quality, security and the computing system's interaction with the environment and the human operator<br>**C5.2 -** Using interdisciplinary knowledge for adapting an information system to application domain requirements<br>**C5.3 -** Using fundamental principles and methods for security, reliability and usability assurance of computing systems<br>**C5.4 -** Adequate utilization of quality, safety and security standards in information processing<br>**C5.5 -** Creating a project including the problem's identification and analysis, its design and development, also proving an understanding of the basic quality requirements |
|---|---|
| 6.2 Cross competences | N/A |

## 7. Discipline objective (as results from the *key competences gained*)

| 7.1 General objective | The overall objective of discipline consists in the study and application of systematic, disciplined and quantifiable approaches in software systems development |
|---|---|
| 7.2 Specific objectives | • Study and application of software development processes<br>• Understanding the specific activities of software engineering<br>• Knowledge of software engineering models<br>• Knowledge of specific tools that can assist software engineers in the specification, design and validation process<br>• Knowledge of methods for software modeling and performance analysis<br>• Application of processes, methods and tools in small to medium-sized software projects |

## 8. Contents

| 8.1 Lectures | Hours | Teaching methods | Notes |
|---|---|---|---|
| Introduction and overview of the course | 2 | PowerPoint presentations, examples, questions, discussion | |
| Software development paradigms: basic paradigms ('waterfall', prototyping, reusable components, formal methods), evolutionary paradigms (incremental development, spiral model, concurrent engineering) | 2 | | |
| Modern processes: the unified process, agile methods and extreme programming | 2 | | |
| Basic activities (specification, development, validation, evolution): concepts, principles, processes | 2 | | |
| Developing requirements: domain analysis, techniques for gathering requirements, capturing requirements as use cases | 2 | | |

| | Hours | | |
|---|:---:|---|---|
| Formal specification: formal modeling and analysis, model checking, tools in support of formal methods (PRISM) | 2 | | |
| Modeling with classes: UML class and object diagrams, using design patterns | 2 | | |
| Modeling with classes: the process of developing class diagrams, semantics of UML class diagrams, implementing class diagrams in Java | 2 | | |
| Modeling interactions and behavior: UML interaction and state diagrams | 2 | | |
| Modeling software behavior: UML state diagrams, software performance modeling and analysis. | 2 | | |
| Architecting and designing software: design principles (increase cohesion, reduce coupling), architectural patterns (Layers, Pipe-and-Filter, etc.) | 2 | | |
| Testing and inspecting to ensure high quality: testing techniques (equivalence partitioning, path testing) and integration strategies (top-down, bottom-up, scenario-based), inspections | 2 | | |
| Use case driven development: use case specifications, analysis, design and implementation to realize the use cases, testing the use cases | 2 | | |
| Program specifications: pre and post assertions, well-founded induction, declarative prototyping | 2 | | |

Bibliography

1. I. Sommerville. *Software Engineering* (6th, 7th, 8th, 9th, 10th editions), Addison Wesley (2001, 2004, 2006, 2010, 2015).
2. T. Lethbridge, R. Laganiere. *Object-Oriented Software Engineering: Practical Software Development using UML and Java* (2nd edition), McGraw-Hill, 2005. http://www.lloseng.com.
3. A. Fox, D. Patterson, *Engineering Software as a Service: An Agile Approach Using Cloud Computing*, Strawberry Canion, 2016.
4. E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1994.
5. E.M. Clarke, T.A. Henzinger, H. Veith, R. Bloem, editors, *Handbook of Model Checking*, Springer, 2018.
6. M. Odersky, L. Spoon, B. Venners, *Programming in Scala,* (3rd, 4th editions), Artima (2016, 2020).
7. E.N. Todoran. *Inginerie software: studii in prototipizare si specificare formala*. Mediamira, Cluj-Napoca, 2006.

| 8.2 Applications – **Laboratory** | Hours | Teaching methods | Notes |
|---|:---:|---|---|
| OCSF – an object client-server framework for reuse oriented development | 2 | | |
| Simple Chat - an instant messaging system based on OCSF (1) | 2 | | |
| Simple Chat - an instant messaging system based on OCSF (2) | 2 | | |
| Using software modeling CASE tools: UML use case, class, interaction, state, component and deployment diagrams | 2 | | |
| Using CASE tools for performance software modeling and analysis: PRISM model checker | 2 | | |
| Using Design Patterns | 2 | | |
| Test cases design, using Junit | 2 | | |
| The **project class** attempts to simulate various aspects of the real world of software engineering. The students define the problem to be solved and the scope of the project under the supervision of the teaching assistant. Working alone is permitted, but they are encouraged to work in teams. The students employ the paradigms and the software development methods that are presented in the taught course, e.g., following the SaaS (Software as a Service) model. They are expected to deliver three iterations of the project with predefined deadlines. For a traditional 'waterfall' project the deadlines correspond to requirements specification, design, and the final deliverable. The project will be delivered in week 13. | 14 | | |
| Bibliography | | | |

1. T. Lethbridge, R. Laganiere. *Object-Oriented Software Engineering: Practical Software Development using UML and Java* (2nd edition). McGraw-Hill, 2005. http://www.lloseng.com.
2. E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1994.
3. PRISM manual, 2020. http://www.prismmodelchecker.org/manual/

*Se vor preciza, după caz: tematica seminariilor, lucrările de laborator, tematica și etapele proiectului.*

## 9. Bridging course contents with the expectations of the representatives of the community, professional associations and employers in the field

Software Engineering is a well-established discipline in Computer Science and Information Technology. In this course, students acquire basic knowledge related to software development (paradigms, methods and tools) and learn to apply systematic and quantifiable approaches in the development of software systems. Course content has been developed based on interaction with specialists in Software Engineering from Romania, Europe (UK, Greece), US and Canada and has been rated by Romanian government agencies (CNEAA and ARACIS).

## 10. Evaluation

| Activity type | Assessment criteria | Assessment methods | Weight in the final grade |
|---|---|---|---|
| Course | Problem solving skills | Written exam | 75% |
| Seminar | | | |
| Laboratory | Software design and validation skills | Laboratory colloquium, | 5% |
| Project | | Project assessment | 20% |

Minimum standard of performance:
Development of a medium size software project using the skills taught in the Software Engineering course.
Grade calculus: 5% laboratory + 20% project + 75% final exam
Conditions for participating in the final exam: Laboratory ≥ 5, Project ≥ 5
Conditions for promotion: grade ≥ 5

| Date of filling in: | Titulari | Titlu Prenume NUME | Semnătura |
|---|---|---|---|
| | Course | Prof.dr.eng. Eneia Todoran | |
| | Applications | Assoc.prof.dr. Paulina Mitrea | |
| | | Assoc.prof.dr.eng. Delia Mitrea | |

| Date of approval in the department | Head of department Prof.dr.ing. Rodica Potolea |
|---|---|
| Date of approval in the Faculty Council | Dean Prof.dr.ing. Liviu Miclea |