Syllabus

Course Description

Course Title	Software Engineering and Project
Course Code	76273
Course Title Additional	
Scientific-Disciplinary Sector	NN
Language	German
Degree Course	Bachelor in Computer Science
Other Degree Courses (Loaned)	
Lecturers	Prof. Dr. Andrea Alexander Janes, Andrea.Janes@unibz.it https://www.unibz.it/en/faculties/engineering/academic-staff/person/2237 Prof. Dr. Claus Pahl, Claus.Pahl@unibz.it https://www.unibz.it/en/faculties/engineering/academic-staff/person/36376
Teaching Assistant	
Semester	First semester
Course Year/s	2
CP	12
Teaching Hours	70
Lab Hours	50
Individual Study Hours	180
Planned Office Hours	36
Contents Summary	Software Engineering: • Software life-cycle: principles and methodologies • Software processes and software project management • Requirements engineering: elicitation and modeling • System modeling and construction: UML, design patterns • Software testing and management: principles and techniques • Recent software engineering topics: DevOps, Cloud, SE and AI

	Coding Capstone: • Project Inception & Roadmap • Architecture & Technical Design • Development & Integration • Deployment & Observability • Final Review & Retrospective • Cross-Team Collaboration & Stakeholder Engagement
Course Topics	Module M1 - Software Engineering – introduces the different phases of a software engineering process. Specific activities, grouped into requirements engineering, systems modelling and design with a focus on architecture concerns, as well as testing and maintenance will be detailed. Recent relevant topics are also discussed. The students will learn the relevant skill in two separate, group-oriented and problem-based projects. Module M2 – Coding Capstone – follows the software lifecycle in a collaborative setting, beginning with a project inception leading into architecture and design. The development and integration phase brings these into programming activities. Deployment and observability ensure that a solution is released and performance monitoring is addressed. The process concludes with a review and retrospective to reflect on successes and identify areas for improvement.
Keywords	Software Engineering, Software Process, Software Architecture, Software Development, Deployment.
Recommended Prerequisites	
Propaedeutic Courses	
Teaching Format	Software Engineering: The Software Engineering course combines frontal lectures with team and/or individual projects. Full-time students work in teams, while non-attending part-time students can opt for individual projects. The projects focus on application developments of a larger scale, addressed in a team- and project-based collaborative work environment. Coding Capstone: The Coding Capstone course is organized as a project-based laboratory in which students work individually or in teams on the design, development, deployment, and evaluation of a software system. The course combines frontal lectures and

	technical briefings with supervised development activities, technical reviews, milestone presentations, and stakeholder discussions, encouraging collaborative work and the practical application of software engineering methodologies in realistic project settings.
Mandatory Attendance	Attendance is not mandatory but strongly recommended. Non-attending students should contact the lecturer at the start of the course to agree on the modalities of the independent study.
Specific Educational Objectives and Learning Outcomes	Knowledge and understanding: - D1.2: Know in details the fundamental principles of programming. - D1.7: To have a thorough knowledge of the main fundamentals techniques and methods of software design, development and maintenance Applying knowledge and understanding: - D2.2: Be able to develop small and medium size programs using different programming languages and paradigms. - D2.3: Be able to solve problems using programming methodologies. - D2.5: Be able to apply the own knowledge to the analysis, design, development and testing of information systems which satisfy given requirements - D2.8: Be able to solve typical problems in computer science based on software engineering methodologies, such as the definition of requirements, the analysis of possible methods for a solution, the selection of the most appropriate methods and tools as well as their application - D2.9: Be able to evaluate the quality of information systems and to identify critical aspects. - D2.16: Be able to coordinate small project teams and to interact with members of the group. Making judgements: - D3.1: Be able to collect and interpret useful data and to judge information systems and their applicability. - D3.2: Be able to work autonomously according to the own level of knowledge and understanding. - D3.3: Be able to take the responsibility for development of projects or IT consulting.

	Communication skills: - D4.1: Be able to use one of the three languages English, Italian and German, and be able to use technical terms and communication appropriately. - D4.2: Be able to negotiate with a customer for the definition of the pre-requisites and features of information systems. - D4.3: Be able to structure and write technical documentation. - D4.4: Be able to work in teams for the realization of IT systems. Learning skills: - D5.1: Have developed learning capabilities to pursue further studies with a high degree of autonomy. - D5.2: Have acquired learning capabilities that enable to carry out project activities in companies, public institutions or in distributed development communities. - D5.3: Be able to follow the fast technological evolution and to learn cutting edge IT technologies and innovative aspects of last generation information systems.
Specific Educational Objectives and Learning Outcomes (additional info.)	
Assessment	The assessment structure differs between the two courses included in this module. Software Engineering is assessed through a project worth 40% and a written exam worth 60%, while Coding Capstone is assessed through a project worth 60% and an oral exam worth 40%. Both attending and non-attending students are required to complete a development project, either individually or in teams. The oral examination assesses each student's theoretical understanding as well as their ability to discuss and reflect on the project results. Students must obtain a passing grade on the project component before being admitted to the oral exam, and both assessment components must be passed in order to successfully complete the module. A positively evaluated project remains valid for three examination sessions. The Software Engineering course provides students with a solid understanding of software life-cycle principles, software processes, requirements engineering, system modeling, design patterns, software testing, and modern topics such as DevOps, Cloud

	computing, and AI-assisted software engineering. These topics contribute to the acquisition of the ability to design, develop, test, and evaluate software systems using appropriate methodologies and tools (D2.2, D2.3, D2.5, D2.8, D2.9). Through modeling activities, documentation, and requirements analysis, students also strengthen their communication and technical writing skills (D4.2, D4.3, D4.4), while the discussion of emerging technologies promotes autonomous learning and continuous professional development (D5.1, D5.3). The Coding Capstone course focuses on the practical realization of software systems through project inception, architecture and technical design, development, deployment, observability, retrospectives, and stakeholder collaboration. By working on realistic development projects, students apply software engineering methodologies in collaborative environments and strengthen their ability to independently design and implement software solutions (D2.2, D2.5, D2.8, D3.2, D3.3). Team coordination, cross-team collaboration, and stakeholder engagement foster communication and teamwork competences (D2.16, D4.1, D4.2, D4.4), while deployment, observability, and retrospective activities develop the ability to evaluate system quality and reflect critically on development processes (D2.9, D3.1). The project-based structure also promotes self-directed learning and prepares students to work effectively in industrial and distributed software development contexts (D5.1, D5.2, D5.3). Both courses contribute to enhance the knowledge in software development and software engineering knowledge (D1.2, D1.7).
Evaluation Criteria	Module 1: Software Engineering For attending students, the grade is calculated based on (i) the lab assessment (40% weight) and (ii) the written final exam (60% weight). For non-attending students, they should follow the delivery schedule for the lab assessments, the grade is calculated the same way. Module 2: Coding Capstone For attending students, the grade is calculated based on (i) the lab assessment (60% weight) and (ii) the written final exam (40% weight). For non-attending students, they should follow the delivery schedule for the lab assessments, the grade is calculated

	the same way. A student needs to be approved in both modules to be approved in the course. The final grade is the average value of the grades from both modules.
Required Readings	Teaching materials, lecture slides, and additional supporting resources will be provided during the course lectures. Software Engineering: Sommerville, I. Software Engineering. 10th Edition. Pearson
Supplementary Readings	
Further Information	
Sustainable Development Goals (SDGs)	Industry, innovation and infrastructure, Quality education

Course Module

Course Constituent Title	Software Engineering and Project M1: Software Engineering
Course Code	76273A
Scientific-Disciplinary Sector	INFO-01/A
Language	German
Lecturers	Prof. Dr. Claus Pahl, Claus.Pahl@unibz.it https://www.unibz.it/en/faculties/engineering/academic-staff/person/36376
Teaching Assistant	
Semester	First semester
CP	6
Responsible Lecturer	
Teaching Hours	40
Lab Hours	20
Individual Study Hours	90
Planned Office Hours	18
Contents Summary	• Software life-cycle: principles and methodologies • Software processes and software project management • Requirements engineering: elicitation and modeling

	• System modeling and construction: UML, design patterns • Software testing and management: principles and techniques • Recent software engineering topics: DevOps, Cloud, SE and AI
Course Topics	
Teaching Format	Frontal lectures, exercises; team and/or individual projects.
Required Readings	• Sommerville, I. Software Engineering. 10th Edition. Pearson.
Supplementary Readings	

Course Module

Course Constituent Title	Software Engineering and Project M2: Coding Capstone
Course Code	76273B
Scientific-Disciplinary Sector	INFO-01/A
Language	German
Lecturers	Prof. Dr. Andrea Alexander Janes, Andrea.Janes@unibz.it https://www.unibz.it/en/faculties/engineering/academic-staff/person/2237
Teaching Assistant	
Semester	First semester
CP	6
Responsible Lecturer	
Teaching Hours	30
Lab Hours	30
Individual Study Hours	90
Planned Office Hours	18
Contents Summary	• Project Inception & Roadmap • Architecture & Technical Design • Development & Integration • Deployment & Observability • Final Review & Retrospective • Cross-Team Collaboration & Stakeholder Engagement
Course Topics	

Teaching Format	The Coding Capstone course is organized as a project-based laboratory in which students work individually or in teams on the design, development, deployment, and evaluation of a software system. The course combines frontal lectures and technical briefings with supervised development activities, technical reviews, milestone presentations, and stakeholder discussions, encouraging collaborative work and the practical application of software engineering methodologies in realistic project settings.
Required Readings	Teaching materials, lecture slides, and additional supporting resources will be provided during the course lectures.
Supplementary Readings	