Syllabus

Course Description

Course Title	Intelligent Tools for Software Evolution
Course Code	76112
Course Title Additional	
Scientific-Disciplinary Sector	INFO-01/A
Language	English
Degree Course	Master in Software Engineering
Other Degree Courses (Loaned)	
Lecturers	
Teaching Assistant	
Semester	Second semester
Course Year/s	1
CP	6
Teaching Hours	40
Lab Hours	20
Individual Study Hours	90
Planned Office Hours	18
Contents Summary	• Introduction to software maintenance and evolution • Software Refactoring • Mining software repositories • Machine learning for software engineering • Assessment and adaptation of intelligent tools for specific maintenance scenarios • Limitations and risks of intelligent tools for software maintenance and their mitigation
Course Topics	Intelligent Tools for Software Evolution explores the modern approaches and technologies that support the maintenance and continuous improvement of software systems. The course begins with an introduction to software maintenance and evolution, emphasizing the challenges of legacy systems, technical debt, and

	the need for sustainable change. Students will learn key refactoring techniques to enhance code quality and maintainability, and explore how software repositories can be mined to extract insights into development patterns, defects, and team behavior. The course integrates machine learning concepts tailored to software engineering tasks, such as bug prediction, code recommendation, and automated documentation. Emphasis is placed on assessing and adapting intelligent tools to fit specific maintenance scenarios, ensuring that solutions are context-aware and effective. Finally, the course addresses the limitations and risks associated with intelligent tools, including issues of trust, explainability, and data bias, and introduces strategies to mitigate these challenges. By the end of the course, students will be equipped to critically evaluate, apply, and adapt intelligent tools to support software evolution in real-world contexts.
Keywords	Software maintenance; Software evolution; Artificial Intelligence applications; Refactoring.
Recommended Prerequisites	
Propaedeutic Courses	
Teaching Format	Frontal lectures, paper presentations, in-class and lab exercises.
Mandatory Attendance	Not compulsory.
Specific Educational Objectives and Learning Outcomes	Knowledge and understanding D1.2 be able to analyse and solve even complex problems in the area of Software Engineering with particular emphasis on the use of empirical evaluation studies, methods, techniques, and technologies. D1.3 have an in-depth knowledge of the scientific method of investigation applied to even complex systems and innovative technologies that support Software Engineering and its various fields of applications. D1.4 have an in-depth knowledge of the principles, structures and use of processing systems for the automation of software systems. D1.8 ability to read, understand, and elaborate on specialist scientific documentation, such as conference proceedings, articles in scientific journals, technical manuals. Applying knowledge and understanding D2.1 know how to apply the fundamentals of data analysis for the

	construction of frameworks, models, techniques, and tools for the evaluation and prediction of characteristics of applications and software systems. D2.3 ability to apply the principles of software engineering to IT and non-IT domains of varying complexity in which software technology is of great importance. D2.5 ability to extend and modify an existing technical solution or theoretical model in an original way, taking into account changing conditions, requirements and the evolution of technology. Making judgements D3.2 ability to plan and re-plan a technical project activity and to carry it out within the defined deadlines and objectives. D3.3 ability to define work objectives compatible with the available time and resources. D3.4 ability to reconcile conflicting project objectives, find acceptable compromises within the limits of cost, resources, time, knowledge, or risk. Communication skills D4.1 ability to present the contents of a scientific/technical report in a set time in front of diverse audiences, including non-specialists. D4.4 ability to prepare and deliver presentations with technical content in English for diverse audiences. D4.6 ability to carry out research and projects in a working group. Learning skills D5.1 ability to independently extend the knowledge acquired during the course of study by reading and understanding scientific and technical documentation in English. D5.3 ability to extend incomplete knowledge with regard to the final objective of the project, in the context of a problem-solving activity.
Specific Educational Objectives and Learning Outcomes (additional info.)	Specific educational objective and learning outcomes (additional information) Software systems can be in use for years, if not decades – extremely prolonged periods during which they must be continuously updated in response to changes in customer needs or other factors. The goal of this course is to teach students basic and advanced techniques to successfully evolve real-world software

	projects. The course will cover the following key software maintenance and evolution activities: • Concept location • Impact analysis • Actualization • Refactoring • Verification The concepts seen during the lecture will be practiced through lab assignment.
Assessment	Student evaluation in the Software Maintenance and Evolution course will be based on lab assignments and a final written exam. The lab assignments will assess students' ability to apply the techniques studied in class to real-world problems (D2.3), demonstrating understanding of empirical methods and software engineering principles (D1.2, D1.3). When assignments involve presenting state-of-the-art papers, students will be evaluated on their ability to read, interpret, and elaborate on scientific literature (D1.8), as well as their capacity to extend and connect that knowledge to other course topics (D5.3). Presentations will also assess the ability to communicate technical content clearly and effectively in English to diverse audiences within set time constraints (D4.1, D4.4), and to contribute collaboratively within project-based learning environments (D4.6). Successful completion of lab work is required to access the final written exam, which will evaluate students' in-depth knowledge of the scientific and technological foundations of software automation (D1.4), data-driven evaluation techniques (D2.1), and their ability to adapt and innovate technical solutions in evolving contexts (D2.5). The exam will also assess planning and judgment skills in aligning objectives with time and resource constraints (D3.2, D3.3, D3.4), as well as clarity of language and critical thinking in summarizing and connecting course topics (D5.1).
Evaluation Criteria	The assessment of the course consists of two parts: • lab assessment, composed of assignments that should be performed and delivered by the due date (50%); • a final written exam (50%).

	In case of a positive mark the lab assessment will count for all 3 regular exam sessions. The lab assignments must be delivered at least one week before the final written exam, otherwise they cannot be assessed, and the exam cannot be registered.
Required Readings	• Vaclav Rajlich, Software Engineering: The Current Practice (Chapman & Hall/CRC Innovations in Software Engineering and Software Development Series). ISBN: 1439841225 • Martin Fowler, Refactoring: Improving the Design of Existing Code (Addison-Wesley Professional). ISBN: 0201485672 • Research papers, that might be recommended by the instructor during the course, will be made available on the course website.
Supplementary Readings	• Robert C. Martin and Michael C. Feathers, Clean code: a handbook of agile software craftsmanship. ISBN: 0136083226
Further Information	
Sustainable Development Goals (SDGs)	Affordable and clean energy, Responsible consumption and production, Industry, innovation and infrastructure, Decent work and economic growth