Fakultät für Ingenieurwesen
Facoltà di Ingegneria
Faculty of Engineering

COURSE DESCRIPTION – ACADEMIC YEAR 2024/2025

Course title	**Software Maintenance and Evolution**
Course code	76064
Scientific sector	INF/01
Degree	Software Engineering for Information Systems (LM-18)
Semester	2
Year	2
Credits	6
Modular	No

Total lecturing hours	40
Total exercise hours	20
Attendance	Not compulsory
Prerequisites	--
Course page	https://ole.unibz.it/

Specific educational objectives	The course belongs to the type "affine o integrative" – and is part of Advanced Topics in Software / Systems Engineering. 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 assignments, based on manipulating large, established open-source software.

Lecturer	Jorge Augusto Melegati Goncalves
Contact	Via Bruno Buozzi 1, Room B1.4.33, jorge.melegati@unibz.it,
Scientific sector of lecturer	INF/01
Teaching language	English
Office hours	By previous appointment by email.
Lecturing Assistant (if any)	--
Contact LA	--
Office hours LA	--
List of topics	• Introduction to software maintenance and evolution • Software Refactoring • Mining software repositories • Machine learning for software engineering • Using software metrics to assess and monitor the quality of software systems • Using textual analysis techniques in the context of software maintenance and evolution

Teaching format	Frontal lectures, paper presentations, in-class and lab exercises

Learning outcomes

Knowledge and understanding

- D1.2 To be able to analyze and solve even complex problems in the area of Software Engineering for Information Systems with particular emphasis on the use of studies, methods, techniques and technologies of empirical evaluation;
- D1.3 To know in depth the scientific method of investigation applied to complex systems and innovative technologies that support information technology and its applications;
- D1.8 To be able to read and understand specialist scientific documentation, such as conference proceedings, articles in scientific journals, technical manuals.

Applying knowledge and understanding

- D2.1 To know how to apply the fundamentals of empirical analysis of ICT data to the construction of mathematical models for the evaluation and prediction of characteristics of applications and software systems;
- D2.3 To know how to apply the principles of software engineering to domains of different complexity, both IT and non-IT, in which software technology is of great importance, such as, for example, in the transport sector or in the medical field;
- D2.5 To be able to extend and modify in an original way an existing technical solution or a formal model taking into account changed conditions, requirements and evolution of the technology.

Making judgments

- D3.2 To be able to plan and re-plan a technical project activity and to carry it out in accordance with defined deadlines and objectives;
- D3.3 To be able to define work objectives compatible with the time and resources available;
- D3.4 To be able to reconcile the objectives of the project that are in conflict, to trade-off cost, resources, time, knowledge or risk.

Communication skills

- D4.1 To be able to present the contents of a scientific/technical report to an audience, including non-specialists, at a fixed time;
- D4.4 To be able to prepare and conduct technical presentations in English;
- D4.6 To be able to carry out research and projects in collaborative manner;
- D4.7 To be able to synthesize knowledge gained from reading and studying scientific documentation.

Learning skills

- D5.1 To be able to independently extend the knowledge acquired during the course of study by reading and understanding scientific and technical documentation in English;
- D5.3 In the context of a problem solving activity, to be able to extend knowledge, even if incomplete, taking into account the final objective of the project.

Assessment	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.
Assessment language	English
Assessment typology	Monocratic commission
Evaluation criteria and criteria for awarding marks	The lab assignments will be assessed based on how students apply the techniques seen in class. Assignments could consist of presentation in class of papers presenting state-of-the-art work in software maintenance and evolution. In this case, they will be assessed based on the understanding of the material presented in the papers, the clarity of the presentation, and the ability to relate it to other topics seen during the course. A positive evaluation on the lab assessment is required to access the written exam. The final written exam will be assessed based on the acquired knowledge and the understanding of the material presented during the course, the clarity of answers, mastery of language (also with respect to teaching language), and the ability to summarize, evaluate, and establish relationships between topics.

Required readings	Lecture slides will be made available on the course website. • 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. Subject Librarian: David Gebhardi, David.Gebhardi@unibz.it
Supplementary readings	• Robert C. Martin and Michael C. Feathers, Clean code: a handbook of agile software craftsmanship. ISBN: 0136083226 Additional resources will be made available on the course website on an as-needed basis.
Software used	The following list includes the most important tools that we will use in the course: • Eclipse DIE, IntelliJ IDEA, or Visual Studio Code • Git • Github • SonarQube