COURSE DESCRIPTION – ACADEMIC YEAR 2019/2020

Course title	**Software Maintenance and Evolution**
Course code	76017
Scientific sector	INF/01
Degree	European Master in Software Engineering (LM-18)
Semester	1
Year	2
Credits	8
Modular	No

Total lecturing hours	48
Total lab hours	24
Total exercise hours	--
Attendance	Not compulsory
Prerequisites	--
Course page	https://ole.unibz.it/

Specific educational objectives	The course belongs to the type "caratterizzanti – discipline "Advanced Topic in Software Engineering" (EMSE - ATSE). Software systems can be in use for years, if not decades – extremely long time periods during which they must be continuously updated to in response to changes in customer needs or other factors. The goal of this course is to teach students basic and advanced techniques in order to successfully evolve real-world software projects. The course will cover the following key software maintenance and evolution activities: o Concept location o Impact analysis o Actualization o Refactoring o Verification The concepts seen during the lecture will be practiced during a project on a large, established open-source software.

Lecturer	Romain Robbes
Contact	Piazza Domenicani 3, Room 1.16, RRobbes@unibz.it, +390471 016025
Scientific sector of lecturer	INF/01
Teaching language	English
Office hours	To be defined and published on the web page of the course.
Lecturing Assistant (if any)	--
Contact LA	--
Office hours LA	--
List of topics	• Introduction to software maintenance and evolution • Supporting maintenance and evolution by mining software repositories • Using software metrics to assess and monitor the quality of software systems • Software Refactoring

	• Using textual analysis techniques in the context of software maintenance and evolution • Using software metrics for bug prediction • Search-based algorithms to support maintenance activities • Release planning: effort and cost estimation techniques
Teaching format	Frontal Lectures, paper presentations, in-class exercises, and group project

Learning outcomes	Knowledge and understanding • Know the main methods and techniques for designing, creating, and maintaining software products and services. • Know the main methods for (re)engineering, refactoring and optimizing software products and processes. • Know the main methods of team, resource management and risks analysis in software development and maintenance. Applying knowledge and understanding • Be able to design and implement information systems in vertical sectors of applications according to technical, functional and organizational requirements. • Be able to apply innovative methods for management and improvement of development processes in different application domains such web or mobile. • Be able to design and execute experimental analyses on information systems or their components. Making judgments • Be able to plan and re-plan a technical project activity aimed at building an information system and to bring it to completion by meeting the defined deadlines and objectives. • Be able to identify reasonable work goals and estimate the resources required to achieve the objectives. Communication skills • Be able to coordinate the work of a project team and to interact positively with members of the group. • Be able to present in a fixed time the content of a scientific / technical report in front of an audience also composed of non-specialists. Learning skills • Be able to autonomously extend the knowledge acquired during the study course by reading and understanding scientific and technical documentation in Italian, German and English. • Be able, in the context of a problem-solving activity, to extend even incomplete knowledge taking into account the objective of the project

Assessment	The assessment of the course consists of three parts: • a project (50%); • oral presentations during the semester (25%) • a final oral exam (25%). In case of a positive mark the project will count for all 3 regular exam sessions. The projects have to be delivered at least one week before

	the final oral 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 project (50% of the mark), in which students implements a series of change requests on an existing, large-scale, open-source software project, applying the techniques seen in class. Furthermore, progress is documented with state of the art tools, and periodically presented in class. The project will be assessed based on how students apply the techniques seen in class during the project, and on how their progress is documented. In case of a group project, each student will be evaluated separately. Papers presenting state of the art work in software maintenance and evolution will be assigned to each students during the semester and will be presented in class. 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. The final oral exam will be assessed based on the acquired knowledge and the understanding of the material presented during the semesters, 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 • Research papers will be made available on the course website and assigned to each course participant
Supplementary readings	• Martin Fowler, Refactoring: Improving the Design of Existing Code (Addison-Wesley Professional). ISBN: 0201485672 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 IDE or IntelliJ IDEA • Git • Github