Fakultät für Informatik
Facoltà di Scienze e Tecnologie informatiche
Faculty of Computer Science

COURSE DESCRIPTION – ACADEMIC YEAR 2016/2017

Course title	**Software Reliability and Testing**
Course code	72095
Scientific sector	INF/01
Degree	Master in Computer Science (LM-18)
Semester	2
Year	1
Credits	8
Modular	No

Total lecturing hours	48
Total lab hours	24
Total exercise hours	--
Attendance	Not compulsory
Prerequisites	Students are familiar with practices and methods of software process management and product and process measurement, are able to develop small software programs in autonomy, and can apply basics of probability and statistics. Pre-requisite material can be taught in the following courses: • Software Process Management • Empirical Software Measurement • Statistical Methods
Course page	https://ole.unibz.it/

Specific educational objectives	The course belongs to the type "caratterizzanti – discipline informatiche" in the curriculum "Software Engineering and IT Management". The course gives a general overview of the scientific contents also tackling advanced techniques both in testing and reliability to prepare the students to their thesis in the subject. It also introduces students to advanced techniques that they can be used in the long term in a research carrier focused on data mining for testing comprehension and generation and reliability prediction.

Lecturer	Barbara Russo
Contact	Piazza Domenicani 3, Room 1.16, barbara.russo@unibz.it, 0471-016170
Scientific sector of lecturer	INF/01
Teaching language	English
Office hours	By previous appointment via e-mail.
Lecturing Assistant (if any)	--
Contact LA	--
Office hours LA	--
Syllabus	• Fundamentals of Software Testing and Analysis • Testing techniques (e.g. Unit and Integration Testing) • Coverage Analysis • System Testing and Test Automation • Test Management • Software reliability and Static Analysis of Software Systems • Dynamic systems and Markov chains

	• Modelling software reliability
Teaching format	Frontal lectures, exercises, and task development solo or in team.

Learning outcomes	Knowledge and understanding • Know the main methods and techniques for designing, creating, and maintaining software products and services. • Understand methods of mathematics and of statistics that support Information Technology and its applications. Applying knowledge and understanding • Be able to apply methods of verification and validation of software • Be able to design and execute experimental analyses on information systems or their components. Ability to make 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 independently select the documentation required to keep abreast of the frequent technological innovations in the field by using a wide variety of documentary sources: books, web, magazines Communication skills • Be fluent, in written and oral form, in at least one European language other than English, with reference also to the specific specialized vocabulary. • Be able to structure and prepare scientific and technical documentation describing project activities Ability to learn • Be able, in the context of a problem-solving activity, to extend even incomplete knowledge taking into account the objective of the project

Assessment	**Knowledge and understanding** The midterm or the written exam contains theoretical open questions on software reliability and testing. Specifically: 1. The students are required to describe a technique and interpret its use in a specific environment. 2. The students are requested to compare techniques and describe their benefits and limitations **Applying knowledge and understanding** At the lab, students will be evaluated through periodic assignments on 1. Their autonomy and competence in selecting and applying testing techniques on actual software. 2. How they use statistical knowledge for fitting models on data and drawing conclusion out of the resulting mathematical findings. 3. How they can design and perform a study with open source software or software publicly available.

	The midterm and the final written exam will contain exercises that builds test suites for given specification or compute probability of defect occurrence. **Making judgments** At the lab, students will be evaluated through periodic assignments on 1. The appropriateness (against given specifications) of testing techniques proposed (verification) 2. The resulting interpretation of the fitting and prediction exercise **Communication skills** At the midterm or at the written exam students will be evaluated by the appropriate use of the course registry. At the lab, students will be evaluated via an assignment that tests their ability to defend their conclusions **Ability to learn** At the lab, students will be evaluated via an intermediate assignment that tests their comprehension of the paradigma behind the single technique and the single model. The assessment is based on assignments (50%) to be done during the semester or all in one at the final exam and a written exam (50%). In the case the assignments are hand in all in one, students are requested to hand in them one week before the final exam date. To access the written exam students must have passed (18 or more) the assignments assessment. In case the assignment assessment is positive but the final written exam is not positive the assignments grade is valid for all three regular exam sessions. There is a midterm. The midterm counts for 50% of the final written exam. In case the midterm grade is positive (18 or greater), the grade is valid for all three regular exam sessions.
Assessment language	English
Evaluation criteria and criteria for awarding marks	*Final grade: 50% lab assessment + 50% written exam* *Lab assessment must be positive (i.e., 18 or higher) to access the written exam.* *Final grade pass: 18 or higher.* Midterm: 50% of the written exam grade. Relevant for the assessment: Lab assessment: ability to apply in autonomy and develop further instruments introduced during the lectures/labs and needed to accomplish tasks and perform little studies in data mining. Ability to report in a professional manner (i.e. using the GQM tool and the guidelines to report empirical studies) also using the appropriate terminology and concepts of the course. Written exam: ability to use the appropriate terminology and concepts of the course and to apply them in different context. Ability to understand the assumptions under which different techniques/methods can better perform or be used.

	Ability to analyze a problem and determine the causes Ability to synthetize the results and interpret them in a specific context also using mathematical instruments to compare and evaluate models shaping software systems in testing or reliability.

Required readings	Lecture notes and papers will be handed out during the course. **Main reference for testing:** Pezzè & Young, Software Testing and Analysis: Process, Principles and Techniques, Wiley, 2007. University Shelf ST 233 P522 . Chap.1-4, 5-6 8-12 17 **Main reference for reliability:** Lyu, M. (ed.) Handbook of Software Reliability Engineering, IEEE Computer Society Press, 1996 Chapter on SRGM **Main reference for Dynamic Systems:** Rigdon E.S. and Basu A.P. Statistical methods for the reliability of repairable systems Wiley series in probability and statistics. Chapter 1-3 **Main reference to review statistic background** Baron, M. Probability and Statistics for computer Scientists Chapmall and Hall, ISBN: 1584886412 University shelf: 15 ST 340 B265(.07). Chapter 1-3 and chapter 6
Supplementary readings	• Paul Ammann and Jeff Offutt, Introduction to Software Testing, Cambridge University Press, Cambridge, UK, ISBN 0-52188-038-1, 2008. • Laurie Williams et al. http://openseminar.org/se/modules/7/index/screen.do • Kent Beck: Test Driven Development by Example, Addison-Wesley Verlag
Software used	• R • Latex • Java, Java script • API to access software repositories In case is needed, students will develop their own tools to mine software reliability data