

SYLLABUS COURSE DESCRIPTION

COURSE TITLE	Software Systems Architecture
COURSE CODE	76225
SCIENTIFIC SECTOR	INF/01
DEGREE	Bachelor in Computer Science
SEMESTER	2nd
YEAR	3rd
CREDITS	6
TOTAL LECTURING HOURS	40
TOTAL LAB HOURS	20
ATTENDANCE	Attendance is not compulsory. Non-attending students have to contact the lecturer at the start of the course to agree on the modalities of the independent study. Exam modalities for non-attending students are the same as for attending students.
PREREQUISITES	-
COURSE PAGE	https://ole.unibz.it/
SPECIFIC EDUCATIONAL OBJECTIVES	<ul style="list-style-type: none"> ● Type of course: caratterizzanti ● Scientific area: discipline informatiche ● To understand the role played by software architecture in software development lifecycle; ● To design software architecture based on patterns and best practices; ● To obtain an overview of different software architecture styles and the newest trends in software architecting; ● To evaluate and balance trade-offs of quality attributes on software architecture; and ● To learn how to apply different software architecture styles to develop high quality software.
LECTURER	Eduardo Martins Guerra
SCIENTIFIC SECTOR OF THE LECTURER	INF/01

TEACHING LANGUAGE	English
OFFICE HOURS	Thursday, 14:00-18:00, Office POS 1.13, eduardo.martinsguerra@unibz.it , +39 375 6071913. Arrange by email.
TEACHING ASSISTANT	-
OFFICE HOURS	-
LIST OF TOPICS COVERED	<ul style="list-style-type: none"> • Software and systems architecture principles • Architecture process and activities: specification, validation • Architectural description and modeling • Stakeholders and viewpoints • Quality considerations: security, performance, modifiability • Patterns of systems architectures
TEACHING FORMAT	Frontal lecture

LEARNING OUTCOMES	<p>Knowledge and understanding:</p> <ul style="list-style-type: none"> • To have a thorough knowledge of the main fundamentals techniques and methods of software design, development and maintenance • Know critical security aspects of information systems, the basic concepts of security and techniques for the development of secure systems. <p>Applying knowledge and understanding:</p> <ul style="list-style-type: none"> • Be able to apply the own knowledge to the analysis, design, development and testing of information systems which satisfy given requirements • 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 <p>Ability to make judgments</p> <ul style="list-style-type: none"> • Be able to collect and interpret useful data and to judge information systems and their applicability • Be able to work autonomously according to the own level of knowledge and understanding <p>Communication skills</p> <ul style="list-style-type: none"> • Be able to structure and prepare scientific and technical documentation • Be able to negotiate with a customer for the definition of the pre-requisites and features of information systems <p>Ability to learn</p> <ul style="list-style-type: none"> • Have developed learning capabilities to pursue further studies with a high degree of autonomy • Be able to independently keep up to date with developments in the most important areas of Computer Science • Have acquired learning capabilities that enable to carry out project activities in companies, public institutions or in distributed development communities
--------------------------	--

ASSESSMENT	<p>The assessment is based on the lab assessment and the final exam. The lab assessment is composed of a number of assignments. The assignments motivate the students to study throughout the semester. The final exam evaluates the students' understanding of the theoretical backgrounds and the ability of solving problems.</p> <p>The students will have the opportunity to perform optional activities that will be evaluated according to the following criteria:</p> <ul style="list-style-type: none"> • Programming challenges, where the student would need to perform a programming task; • Tool or technology research, where the student would need to search information about a tool or technology; • Quests, where the student will be challenged to search for real examples in local companies; • Judgements, where students will judge a technology or technique presenting both positive and negative points; • Participation and performance in educational games and quizzes performed in the classroom (just for attending students).
ASSESSMENT LANGUAGE	<p>English</p>
EVALUATION CRITERIA AND CRITERIA FOR AWARDING MARKS	<p>For both, attending and non-attending students, the grade is calculated based on (i) the lab assessment (40% weight) and (ii) the final exam (60% weight).</p> <p>Student optional activities will be used for a score in a Gamification system. Based on students' position on ranking, they might get some bonus in their final grade.</p>
REQUIRED READINGS	<p>Robert C. Martin. 2017. Clean Architecture: A Craftsman's Guide to Software Structure and Design (1st ed.). Prentice Hall Press, Upper Saddle River, NJ, USA.</p> <p>Mark Richards. 2015. Software Architecture Patterns. O'Reilly Media, Inc..</p>
SUPPLEMENTARY READINGS	<p>Len Bass, Paul Clements, and Rick Kazman. 2012. Software Architecture in Practice (3rd ed.). Addison-Wesley Professional.</p>
SOFTWARE USED	<p>Java JDK, Eclipse (or other Java IDE), other open-source tools.</p>