Syllabus

Course Description

Course Title	Software Systems Engineering
Course Code	76265
Course Title Additional	
Scientific-Disciplinary Sector	
Language	English; Italian
Degree Course	Bachelor in Computer Science
Other Degree Courses (Loaned)	
Lecturers	Prof. Barbara Russo, Barbara.Russo@unibz.it https://www.unibz.it/en/faculties/engineering/academic-staff/person/2242 Prof. Eduardo Martins Guerra, Eduardo.MartinsGuerra@unibz.it https://www.unibz.it/en/faculties/engineering/academic-staff/person/43879
Teaching Assistant	
Semester	Second semester
Course Year/s	2
CP	12
Teaching Hours	70
Lab Hours	50
Individual Study Hours	180
Planned Office Hours	
Contents Summary	- Software and systems architecture principles - Software architecture design process - Architectural components and frameworks - Approaches for architectural partitioning - Architectural patterns and styles - Integrating AI Components into Architectural Designs

	- Verification and validation - Techniques for black-box and white-box texting - Techniques and tools for test automation - Integration and regression testing - Web testing - AI for test case generation
Course Topics	The objective of the Software Systems Architecture Module is to provide students with a solid understanding of the role of software architecture within the software development lifecycle. Students will learn to design architectures using established patterns and best practices, gain an overview of various architectural styles and emerging trends, evaluate and balance trade-offs related to quality attributes, and apply different architectural approaches to the development of high-quality software. Tools and Techniques for Software Testing is designed to equip students with the ability to select, use, customize, and deploy software testing tools and techniques. It also enables them to configure and integrate these tools to support testing activities throughout the collaborative software development process.
Keywords	Software testing, prompt engineering for testing, agentic systems for testing
Recommended Prerequisites	Basic courses in Programming and Software Engineering. Familiarity with UML and the basics of object-oriented programming.
Propaedeutic Courses	
Teaching Format	The course includes frontal lectures and lab exercises.
Mandatory 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.
Specific Educational Objectives and Learning Outcomes	Knowledge and Understanding - D1.8 To have a thorough knowledge of the main fundamentals techniques and methods of software design, development and maintenance Applying knowledge and understanding - D2.5 Be able to apply the own knowledge to the , design, development and testing of information systems which satisfy

	given requirements - D2.7 Be able to conduct simple experiments about information systems by collecting measures about the behaviour of the system. - D2.10 Be able to solve typical problems in computer science based on software engineering methodologies, such as the definition of requirements, the of possible methods for a solution, the selection of the most appropriate methods and tools as well as their application - D2.11 Be able to evaluate the quality of information systems and to identify critical aspects. Ability to make judgments - D3.1 Be able to collect and interpret useful data and to judge information systems and their applicability. - D3.2 Be able to work autonomously according to the own level of knowledge and understanding. - D3.3 Be able to take the responsibility for development of projects or IT consulting. Communication skills - D4.1 Be able to use one of the three languages English, Italian and German, and be able to use technical terms and communication appropriately. - D4.3 Be able to negotiate with a customer for the definition of the pre-requisites and features of information systems. - D4.4 Be able to structure and write technical documentation. Learning skills - D5.1 Have developed learning capabilities to pursue further studies with a high degree of autonomy. - D5.2 Have acquired learning capabilities that enable to carry out project activities in companies, public institutions or in distributed development communities. - D5.3 Be able to follow the fast technological evolution and to learn cutting edge IT technologies and innovative aspects of last generation information systems.
Specific Educational Objectives and Learning Outcomes (additional info.)	

Assessment	A final unified written exam will be performed with verification questions from both modules. Each module will have the following additional activities during the course: Software Systems Architecture: Lab assessment is composed by assignments that should be performed and delivered in each week. Optional activities might worth extra points in the final module grade. Tools and Techniques for Software Testing: Project work with group assignments
Evaluation Criteria	The written exam evaluates the student's ability to master the course terminology, assess tools and techniques in relation to their specific domain of use and technical details, solve exercises, and clearly summarize theoretical concepts. The assessment of project and lab work focuses on timely completion of assignments, the delivery of functional solutions, and the development or customization of high-quality software that meets the specified requirements. The final grade is calculated as the average of the grades obtained in each module. For Software Systems Architecture, the grade is based on two components: 50% from lab work and 50% from the module-specific questions in the final written exam. Optional activities within the module may provide extra credit. For Tools and Techniques for Software Testing, the grade is composed of 80% from project work and 20% from the relevant questions in the written exam. Completion and positive evaluation of the project is a prerequisite for accessing the written exam. For non-attending students, module instructors may offer two options: either complete lab assessments or projects under the same conditions as attending students, or undergo an additional evaluation based on questions related to lab or project content. Students must pass both modules to pass the course. A positive grade in one module remains valid for all three regular exam sessions within the academic year.

Required Readings	• Robert Martin. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Pearson, London, England, 1st edition, September 2017. ISBN 978-0-13-449416-6. • Mark Richards. Software Architecture Patterns. O'Reilly Media, Inc., 2015. ISBN 978-1-4919-2540-9. • Paul C. Jorgensen, Software Testing: A Craftsman's Approach, Fourth Edition, CRC Press, 2013 ISBN 1466560681 • Mauricio Anniche Effective Software Testing: A developer's guide ¿ 2022 Manning Publications ISBN 9781633439931 https://unibz.primo.exlibrisgroup.com/discovery/fulldisplay?docid=cdi_proq
Supplementary Readings	Len Bass, Paul Clements, and Rick Kazman. Software Architecture in Practice. Addison-Wesley Professional, Harlow, 3rd edition, September 2012. ISBN 978-0-321-81573-6.
Further Information	- Java (https://openjdk.org) - A Java IDE (e.g., https://eclipseide.org) - Git (https://git-scm.com) - Maven (https://maven.apache.org) - Issue trackers - JaCoCo (https://www.eclemma.org/jacoco/) - FindBugs (https://findbugs.sourceforge.net) - JUnit 5 (https://junit.org) - FitNesse (https://fitnesse.org)
Sustainable Development Goals (SDGs)	Quality education

Course Module

Course Constituent Title	Software Systems Architecture
Course Code	76265A
Scientific-Disciplinary Sector	IINF-05/A
Language	English
Lecturers	Prof. Eduardo Martins Guerra, Eduardo.MartinsGuerra@unibz.it https://www.unibz.it/en/faculties/engineering/academic-staff/person/43879
Teaching Assistant	
Semester	Second semester

CP	6
Responsible Lecturer	
Teaching Hours	40
Lab Hours	20
Individual Study Hours	90
Planned Office Hours	
Contents Summary	- Software and systems architecture principles - Software architecture design process - Architectural components and frameworks - Approaches for architectural partitioning - Architectural patterns and styles - Integrating AI Components into Architectural Designs
Course Topics	The objective of the Software Systems Architecture Module is to provide students with a solid understanding of the role of software architecture within the software development lifecycle. Students will learn to design architectures using established patterns and best practices, gain an overview of various architectural styles and emerging trends, evaluate and balance trade-offs related to quality attributes, and apply different architectural approaches to develop high-quality software.
Teaching Format	The course includes frontal lectures and lab exercises.
Required Readings	Robert Martin. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Pearson, London, England, 1st edition, September 2017. ISBN 978-0-13-449416-6. Mark Richards. Software Architecture Patterns. O'Reilly Media, Inc., 2015. ISBN 978-1-4919-2540-9.
Supplementary Readings	Kyle Brown, Bobby Woolf, and Joseph Yoder. Cloud Application Architecture Patterns: Designing, Building, and Modernizing for the Cloud, 2025, Oreilly & Associates Inc, ISBN 978-1098116903

Course Module

Course Constituent Title	Tools and Techniques for Software Testing
Course Code	76265B
Scientific-Disciplinary Sector	INFO-01/A
Language	Italian
Lecturers	Prof. Barbara Russo, Barbara.Russo@unibz.it https://www.unibz.it/en/faculties/engineering/academic-staff/person/2242
Teaching Assistant	
Semester	Second semester
CP	6
Responsible Lecturer	
Teaching Hours	30
Lab Hours	30
Individual Study Hours	90
Planned Office Hours	
Contents Summary	- Verification and validation - Techniques for black-box and white-box texting - Techniques and tools for test automation - Integration and regression testing - Web testing - AI for test case generation
Course Topics	Tools and Techniques for Software Testing is designed to equip students with the ability to select, use, customize, and deploy software testing tools and techniques. It also enables them to configure and integrate these tools to support testing activities throughout the collaborative software development process and agentic systems.
Teaching Format	The course includes frontal lectures and lab exercises.
Required Readings	· Paul C. Jorgensen, Software Testing: A Craftsman's Approach,

	Fourth Edition, CRC Press, 2013 ISBN 1466560681 · Mauricio Anniche Effective Software Testing: A developer's guide ¿ 2022 Manning Publications ISBN 9781633439931 https://unibz.primo.exlibrisgroup.com/discovery/fulldisplay?docid=cdi_proquest_e
Supplementary Readings	