Syllabus

Course description

Course title	Software Design and Implementation
Course code	76105
Course title additional	
Scientific sector	INF/01
Teaching language(s)	English
Degree course	Master in Software Engineering
Other degree courses (loaned)	
Lecturer(s)	Prof. Dr. Claus Pahl, Claus.Pahl@unibz.it https://www.unibz.it/it/faculties/engineering/academic-staff/person/36376-claus-pahl Dr. Eduardo Martins Guerra, eduardo.guerra@unibz.it https://www.unibz.it/it/faculties/engineering/academic-staff/person/43879-eduardo-martins-guerra
Teaching assistant(s)	
Semester	1
Course year	1
CP	12
Teaching hours	180
Lab hours	40
Individual study	80
Planned office hours	
Contents summary	Module M1 - Requirements Engineering for Dependable Systems • Functional and Non-Functional Requirements • Requirements Engineering Processes • Requirements Elicitation and Analysis

	• Requirements Specification • Validation of Requirements • Requirements Change Module M2 - Software Architecture • Quality Attributes and Software Architecture Concepts • Architecture Partitioning (layers, modules, components) • Flexible and Adaptive Architectural Design • Architectural Patterns and Styles • Integrating AI Components into Architectural Designs • Continuous Architecture
Course content	Module M1 - Requirements Engineering for Dependable Systems – defines different types of requirements and introduces the different phases of a requirements engineering process. This provides a generic process framework. In the second part of this module, the focus is on dependable systems and specific requirement types and processes for this context, addressing in particular metrics for software quality. The students will learn the relevant skill in two separate, group-oriented and problem-based projects. Module M2 - Software Architecture - This course explores the foundational concepts of software architecture, emphasizing quality attributes and architectural design principles. It covers architecture partitioning through layers, modules, and components, and focuses on creating flexible and adaptive systems. Students will examine key architectural patterns and styles, learn strategies for integrating AI components into system architectures, and understand the principles of continuous architecture to support ongoing system evolution and improvement.
Keywords	Requirements Engineering, Dependability Requirements, Software Architecture, Software Design
Prerequisites	Basic courses in Programming and Software Engineering. Familiarity with UML and software modelling. Familiarity with the basics of Object-orientation and automated testing.

Propaedeutic courses	N/A
Teaching format	Frontal lectures, exercises; team and/or individual projects.

Mandatory attendance	Not compulsory. Non-attending students must 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 D1.1 possess solid knowledge of both the fundamentals and the application aspects of the various fundamental areas of computer science; D1.2 be able to analyse and solve even complex problems in the area of Software Engineering for Information Systems with particular emphasis on the use of empirical evaluation studies, methods, techniques, and technologies; D1.3 have an in-depth knowledge of the scientific method of investigation applied to even complex systems and innovative technologies that support Software Engineering and its various fields of applications. D1.4 have an in-depth knowledge of the principles, structures and use of processing systems for the automation of software systems. D1.5 know the fundamentals, techniques, and methods of design, customisation and implementation of software to support the automation of new-generation software systems for industrial production, company business, education, and society. D1.6 understand the elements of corporate and professional culture. D1.7 know the various fields of application of Software Engineering also with reference to the local, national, and international economic-social context. Applying knowledge and understanding D2.3 ability to apply the principles of software engineering to IT and non-IT domains of varying complexity in which software technology is of great importance. D2.4 ability to define an innovative technical solution to an application problem that respects technical, functional, and organisational constraints and requirements. D2.5 ability to extend and modify an existing technical solution or theoretical model in an original way, taking into account changing conditions, requirements and the evolution of technology. Making judgments D3.2 ability to plan and re-plan a technical project activity and to carry it out within the defined deadlines and objectives. D3.3 ability to define work objectives compatible with the available time and resources. D3.4 ability to reconcile conflicting project objectives, find acceptable compromises within the limits of cost, resources, time, knowledge, or risk. D3.5 ability to work with broad autonomy, taking responsibility for projects and structures.

	Communication skills D4.2 ability to structure and draft scientific and technical descriptive documentation of project activities for diverse audiences. D4.3 ability to work and co-ordinate the work of a multi-disciplinary project team, to identify activities aimed at achieving the project objectives. D4.5 ability to interact and collaborate in the realisation of a project or research with peers and experts. Learning skills D5.2 ability to independently keep up to date with developments in the most important fields of information technology.

Specific educational objective and learning outcomes (additional information)	Module 1: Requirements Engineering for Dependable Systems The course objective is to familiarize students with advanced techniques and tools to elicit, specify and manage software system requirements, aiming to understand both conceptual foundations as well as practical applicability. The students will acquire skills to elicit requirements in various settings and specify them in a way that permits communication with various stakeholders, but also suitable for managing change in software projects. Quality management is specifically introduced. The students are exposed to problem-solving skills that allow requirements engineering in a dynamic, multi-stakeholder setting. Module 2: Software Architecture The following are the module specific objective: 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; to design architectures that integrate artificial intelligence components into applications; and to learn how to apply different software architecture styles to develop high quality software.
Assessment	Module 1: Requirements Engineering for Dependable Systems The assessment is based on the lab assessment and the final written exam. The lab assessment is composed practical activities that can be performed by the students during the course. The final written exam evaluates the students' understanding of the theoretical backgrounds and the ability of solving problems. The student should achieve at least 50% of the lab grade to do the final exam. Module 2: Software Architecture

	The assessment is based on the lab assessment and the final written exam. The lab assessment is composed of practical activities that can be performed by the students during the course. The final written exam evaluates the students' understanding of the theoretical background and the ability to solve problems. The student should achieve at least 50% of the lab grade to do the final exam. The written exam will evaluate the student's knowledge (D1.1, D1.2, D1.3, D1.4, D1.5, D1.6, D1.7) and how this knowledge can be applied to specific problems (D2.3, D2.4, D2.5). The course labs and activities will evaluate their decision-making capacity in the context of software projects (D3.2, D3.3, D3.4, D3.5), exercising their communication skills (D4.2, D4.3, D4.5). Learning skills will be evaluated in practical activities, in which students need to research new technologies and methods (D5.2) in the context of each module.
Evaluation criteria	Module 1: Requirements Engineering for Dependable Systems For attending students, the grade is calculated based on (i) the lab assessment (50% weight) and (ii) the written final exam (50% weight). For non-attending students, they should follow the delivery schedule for the lab assessments, the grade is calculated the same way. Module 2: Software Architecture For attending students, the grade is calculated based on (i) the lab assessment (50% weight) and (ii) the written final exam (50% weight). For non-attending students, they should follow the delivery schedule for the lab assessments, the grade is calculated the same way. A student needs to be approved in both modules to be approved in the course. The final grade is the average value of the grades from both modules.
Required readings	Sommerville, I. (2015). Software Engineering. 10th Edition. Pearson. Laplante, P.A., and Kassab, M.H. (2022). Requirements Engineering for Software and Systems. CRC Press. 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. Mark Richards. 2015. Software Architecture Patterns. O'Reilly Media, Inc.

Supplementary readings	Johnson, R., & Vlissides, J. (1995). Design patterns. Elements of Reusable Object-Oriented Software Addison-Wesley, Reading. Fowler, M. (2018). Refactoring: improving the design of existing code. Addison-Wesley Professional. Evans, E., & Evans, E. J. (2004). Domain-driven design: tackling complexity in the heart of software. Addison-Wesley Professional. Len Bass, Paul Clements, and Rick Kazman. 2012. Software Architecture in Practice (3rd ed.). Addison-Wesley Professional. Open educational resources, representing alternative or supplementary materials, shall be linked to the course website.

Further information	Software Modelling (e.g., Argo UML, Papyrus, StarUML, draw.io), Java JDK, Java Programming IDE (e. g. Eclipse, Intellij)
Sustainable Development Goals (SDGs)	Affordable and clean energy + industry, innovation and infrastructure + sustainability cities and communities + responsible consumption and production

Course module

Course constituent title	Software Design and Implementation M1 - Requirements Engineering for Dependable Systems
Course code	76105A
Scientific sector	INFO-01/A
Teaching language(s)	English
Lecturer(s)	Prof. Dr. Claus Pahl, Claus.Pahl@unibz.it https://www.unibz.it/it/faculties/engineering/academic-staff/person/36376-claus-pahl
Teaching assistant(s)	
Semester	1
CP	6
Responsible lecturer	
Teaching hours	40
Lab hours	20
Individual study	90
Planned office hours	18

Contents summary	• Functional and Non-Functional Requirements • Requirements Engineering Processes • Requirements Elicitation, Analysis, Specification, Verification • Dependability Systems Principles • Dependability Requirements and Metrics • Requirements Change

Course content	The course objective is to familiarize students with advanced techniques and tools to elicit, specify and manage software system requirements, aiming to understand both conceptual foundations as well as practical applicability. The students will acquire skills to elicit requirements in various settings and specify them in a way that permits communication with various stakeholders, but also suitable for managing change in software projects. Quality management is specifically introduced. The students are exposed to problem-solving skills that allow requirements engineering in a dynamic, multi-stakeholder setting. The first part will focus on generic requirements engineering. The second part will deepen dependable systems requirements and respective techniques
Teaching format	Frontal lectures, exercises; team and/or individual projects.
Required readings	Sommerville, I. (2015). Software Engineering. 10th Edition. Pearson. Laplante, P.A., and Kassab, M.H. (2022). Requirements Engineering for Software and Systems. CRC Press.
Supplementary readings	Open educational resources, representing alternative or supplementary materials, shall be linked to the course website.

Course module

Course constituent title	Software Design and Implementation M2 - Software Architecture

Course code	76105B
Scientific sector	ING-INF/05
Teaching language(s)	English
Lecturer(s)	Dr. Eduardo Martins Guerra, eduardo.guerra@unibz.it https://www.unibz.it/it/faculties/engineering/academic-staff/person/43879-eduardo-martins-guerra

Teaching assistant(s)	
Semester	1
CP	6
Responsible lecturer	
Teaching hours	40
Lab hours	20
Individual study	90
Planned office hours	135
Contents summary	• Quality Attributes and Software Architecture Concepts • Architecture Partitioning (layers, modules, components) • Flexible and Adaptive Architectural Design • Architectural Patterns and Styles • Integrating AI Components into Architectural Designs • Continuous Architecture
Course content	This course provides a comprehensive exploration of foundational and advanced topics in software architecture, focusing on both theoretical understanding and hands-on application. Students will begin by examining key quality attributes and essential software architecture concepts, followed by strategies for architecture partitioning, including the use of layers, modules, and components. Emphasis will be placed on flexible and adaptive architectural design to accommodate evolving requirements. The course also covers a range of architectural patterns and styles, empowering students with tools to make informed design decisions. A modern perspective is introduced through the integration of AI components into architectural designs, preparing students to address current industry demands. Additionally, the concept of continuous architecture will be explored to support ongoing system evolution. Throughout the course, students will engage in practical activities that reinforce theoretical knowledge and promote the application of architectural principles in real-world scenarios.
Teaching format	Frontal lectures, exercises; team and/or individual projects.
Required readings	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. Mark Richards. 2015. Software Architecture Patterns. O'Reilly Media, Inc.

Supplementary readings	Johnson, R., & Vlissides, J. (1995). Design patterns. Elements of Reusable Object-Oriented Software Addison-Wesley, Reading. Fowler, M. (2018). Refactoring: improving the design of existing code. Addison-Wesley Professional. Evans, E., & Evans, E. J. (2004). Domain-driven design: tackling complexity in the heart of software. Addison-Wesley Professional. Len Bass, Paul Clements, and Rick Kazman. 2012. Software Architecture in Practice (3rd ed.). Addison-Wesley Professional. Open educational resources, representing alternative or supplementary materials, shall be linked to the course website.