COURSE DESCRIPTION – ACADEMIC YEAR 2021/2022

Course title	**Advanced Software Design Techniques**
Course code	76087
Scientific sector	INF/01
Degree	Master in Software Engineering for Information Systems (LM-18)
Semester	1
Year	1
Credits	6
Modular	No

Total lecturing hours	40
Total exercise hours	20
Attendance	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.
Prerequisites	Basic courses in Programming and Software Engineering. Familiarity with UML and software modelling. Familiarity with the basics of object-orientation and automated testing.
Course page	https://ole.unibz.it/

Specific educational objectives	The course belongs to the type caratterizzanti – discipline informatiche and is part of the Foundations in Software Engineering. The course objective is to familiarize students with advanced tools and techniques to design a software-based system, understanding in theory and practice their applicability to achieve software requirements and the expected consequences in the implementation of each one. Students will acquire skills and competencies resulting from identifying requirements and implementing them through various design techniques, focusing on improving code maintenance and introducing extensibility in the suitable spots. Design evaluation is approached through the analysis of metrics and visualization techniques. The students are exposed to problem-solving techniques that allow the synthesis of software design solutions satisfying the system's requirements.

Lecturer	Martins Guerra Eduardo
Contact LA	Piazza Domenicani 3, eduardo.guerra@unibz.it
Scientific sector of lecturer	ING-INF/05
Teaching language	English
Office hours	During the lecture times, and by arrangement by email
Lecturing Assistant (if any)	--
Contact LA	--
Office hours LA	--
List of topics	• Design Patterns Application and Interaction • Evolutionary Design Techniques (TDD, BDD, Refactoring) • Domain Modeling (DDD) • Components and Modularization • Framework Development (Extension Points, Reflection, Metadata)

	• Software Design Evaluation (Code Metrics, Code Smells, Software Visualization)
Teaching format	Frontal lectures, exercises; team and/or individual projects.

Learning outcomes	**Knowledge and understanding** • D1.2 To be able to analyze and solve even complex problems in the area of Software Engineering for Information Systems with particular emphasis on the use of studies, methods, techniques and technologies of empirical evaluation; • D1.4 To know in depth the principles, structures and use of computer systems for the automation of information systems; • D1.5 To know the fundamentals, techniques and methods of design, customization and implementation of software to support the automation of new generation information systems for industrial production and business; **Applying knowledge and understanding** • D2.4 To be able to define an innovative technical solution to an application problem that meets technical, functional and organisational constraints and requirements; • D2.5 To be able to extend and modify in an original way an existing technical solution or a formal model taking into account changed conditions, requirements and evolution of the technology; **Making judgments** • D3.4 To be able to reconcile the objectives of the project that are in conflict, to trade-off cost, resources, time, knowledge or risk; **Communication skills** • D4.5 To be able to prepare and conduct technical presentations in English; • D4.6 To be able to interact and collaborate during the implementation of a project or research with peers and experts; **Learning skills:** • D5.2 To be able to keep up to date independently with developments in the most important areas of information technology; • D5.3 In the context of a problem solving activity, to be able to extend knowledge, even if incomplete, taking into account the final objective of the project;

Assessment	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. The students will have the opportunity to perform optional activities during the course, such as:

	• Programming challenges, where the student would need to perform a programming task • Tool or technology research, where the students 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. The optional activities completed will score points in a gamification system that will be used to give bonus points for the students in their final grade.
Assessment language	English
Assessment typology	Monocratic
Evaluation criteria and criteria for awarding marks	For attending students, the grade is calculated based on (i) the lab assessment (50% weight) and (ii) the final exam (50% weight). 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. For non-attending students, if they can follow the delivery schedule for the lab assessments, the grade is calculated the same way. Unless, the grade is calculated based only on the final exam that will include questions related to the labs' content.

Required readings	The course will be based on lecture notes.
Supplementary readings	• Johnson, R., & Vlissides, J. (1995). Design patterns. *Elements of Reusable Object-Oriented Software Addison-Wesley, Reading*. • Beck, K. (2003). Test-driven development: by example. Addison-Wesley Professional. • 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. • Lanza, M., & Marinescu, R. (2007). *Object-oriented metrics in practice: using software metrics to characterize, evaluate, and improve the design of object-oriented systems*. Springer Science & Business Media. • Open educational resources, representing alternative or supplementary materials, shall be linked to the course website.
Software used	Software Modelling (e.g. Argo UML, Papyrus, StarUML), Java JDK, Java Programming IDE (e. g. Eclipse, Intellij)