Study programme(s): Informatics (IM)

Level: master

Course title: Architecture, Design and Patterns (code IB122)

Lecturers: Vladimir Kurbalija, Danijela N. Boberić-Krstićev

Status: obligatory for the *Software engineering* module; elective for other modules.

ECTS: 7,5

Requirements: none

Learning objectives

To introduce students to a multitude of modelling techniques and designs to address the issue of software architecture in the context of object-oriented software development. Course covers all aspects of software design from architectural features (styles, models and views) to design models that could be described as "a common solution to common problems in a given context" on the lower level of abstraction.

Learning outcomes

Minimum: Students are expected to show understanding of the impact of abstraction, modelling, architecture, and patterns in software product development and be able to critically discuss software architectures, the key concepts, designs and patterns.

Optimal: Students are expected to be able to critically discuss the architectural alternatives and alternative designs, to generate reasonable alternatives for the problem and select between them, to identify an appropriate pattern for the problem and create it, and to apply practical skills in generating and developing software architecture and design based on functional requirements.

Syllabus

Theoretical instruction

Theoretical background of software architecture, analogy with architecture in general, the elements of software architecture, architectural styles (ABAS), architectural patterns (Event-based, Layered, Pipes & Filters, ...), architecture description languages, the interaction between requirements and architecture, master-plan vs. piecemeal growths, architecture analysis and evaluation (SAAM, Scenario-based evaluation), the architectural process and organization, model driven architecture, from the architecture to the model, re-usable architecture, design patterns, framework and tools.

Practical instruction

Case study analysis.

Literature

1. Len Bass, Rick Kazman, Paul Clements, Software Architecture in Practice, Addison Wesley, second edition.
2. M. Shaw and D. Garlan, Software Architecture. Prentice Hall 1996
3. Erich Gamma, Richard Helm, Ralph Johnson, John M. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison Wesley.
4. H. Rumbaugh, M. Blaha, W. Premarlani, F. Eddy, W. Lorensen: Object-Oriented Modelling and Design, Prentice-Hall
5. G. Booch: Object-Oriented Analysis and Design with Applications, Addison-Wesley, 1994 (2nd ed.)

Weekly teaching load				Other:
Lectures: 3	Exercises: 2	Other forms of teaching:	Student research:	

Teaching methodology

Classical methodology is applied in lectures including the use of video-beam and slides projector. During the exercises, the case studies and examples are analyzed by using the traditional methods of teaching and using the video-beam. Practical skills in architecture and modelling are developed with the introduction of the recommended tools. Students gradually build their knowledge in each research topic. Knowledge is checked through the realization of projects that are presented during and at the end of the course.			
Grading (maximum number of points 100)			
Pre-exam obligations	**points**	**Final exam**	**points**
Practical instruction	**10**	Oral exam	40
Seminar(s)	**50**		