Course title: SOFTWARE ANALYSIS AND DESIGN

Lecturers	Assoc. Prof. Zlatko Stapić, Ph. D. Full Prof. Neven Vrček, Ph.D. Marko Mijač, Ph.D. Dijana Peras, M.Inf. Žana Zekić, M.Inf.
Language of instruction	Croatian and English
Study level	Master
Study programme	Information and Software Engineering
Semester	1st (winter)
ECTS	6
Goal	The goal of the course software analysis and design is to introduce the students to the life cycle and development phases of a modern software product with the emphasis on architectural design and implementation of mobile software products. Software development has become an important branch of industry which has its patterns and related standards. The course deals with all the phases of the software development life cycle, which the new software product must undergo at the beginning of its creation: analysis of the system's domain, specification of the software requirements, methods and techniques of software modelling, software development, software testing and removal of errors. In this way students learn about basic approaches used in development and engineering of complex, software-based systems, and they also learn about modern tools that facilitate software development and steps of software development lifecycle.
General and specific learning outcomes	
Content	**1. Paradigms of software systems' development life cycle** Life cycle of a software product. Approaches to the development of software system and possible variations: waterfall, spiral, agile. Complex development cycles (parallel development, feedback). **2. Software system development lifecycle** Project specificities in software industry. Relationship between the project and the development cycle of a software product. Characteristic methods of planning and tracking the project in development of a software product. Expenses of a product. Project teams and their characteristics: specialization areas, required knowledge, overlapping of the knowledge areas. Virtual project teams and tools that support group work (teamwork, groupware). **3. Analysis of software system requirements – users' requirements** Definition of user's requirements. Business processes and influence on users' requirements. Sources of users' requirements. Organizing the users' requirements. Techniques of gathering the users' requirements: interviews, inquiries, business documents, ... **4. Analysis of software system requirements – system requirements**

	Definition of system requirements. Types of system requirements. Mapping between user and system requirements. Functional and non-functional requirements. Transition and dynamic modelling. Organizing and documenting of functional and non-functional requirements. **5. Software system modelling** Software system architecture and basic construction elements. Software modelling diagram techniques. Standards and approaches to modelling of software system. Basic concepts of OO approach. Inheriting, encapsulation, polymorphism. Object-oriented approach in program languages and tools. **6. UML paradigm** UML diagrams and their use in design phase of software development. **7. Concepts of software product development design** Definition of the software product development design. Basic questions of architecture development (e.g. data requirements, managing the memory, exceptions, etc.). Design principles (hiding the information, cohesion and pairing). Interactions between the design, functional and non-functional requirements. Design oriented to quality of the attributes (ex. reliability, usability, performance, possibilities of testing, tolerance of errors, etc.). Architectural styles, reusability. Interoperability. SOLID design principles. **8. Tools for development and modelling of software systems** Types of tools for development and modelling of software systems (ex. architectural, for static analysis, for dynamic estimation, etc.). Typical tool architectures. Possibilities and limitations of tools. **9. Software system architecture** Layers of software system architecture and typical architectures. Characteristic technologies in each of the layers. Connecting the layers and integration of the software system. Influence of the architecture on characteristics of the software system (resistance to incidents, malfunctions, speed). **10. Components and integration** Component paradigm. Reusability of the program code. Types of software components. Technologies for the development and integration of software components. Managing the transactions of the components. Integration of the components. Market of the software components. **11. User interface and user experience** General principles of the design of human computer interface. Psychology of the human computer interface. Basic elements of visual design (e.g. colors, icons, letter types, etc.). Reply time and feedback information. Design approaches (e.g. Oriented to me, the forms, questions-answers, etc.); Localization and internationalization. Advanced design methods of the human computer interface. Design of augmented virtual reality. Metaphors and conceptual models. **12. Prototyping** Purpose of the prototype in software industry. Types of prototype: horizontal and vertical. Prototype planning. Documenting the prototype. Testing scenarios. Relation between the prototype and the real system. Software tools for prototype development.

	13. Metrics in the software development The principles of software metrics and their applicability. Types of metrics: lines of program code, functional points. Metrics and life cycle. Methods for estimating complexity of software system. Static and dynamic code analysis. **14. Software testing** Significance and approaches to testing of software system. Testing of the components and the entire integrated software system. Relation of software performance and users' requirements. Testing scenarios. Analysis of the range of questioning (e.g. branch, basic course, multiple conditions, data flow, exceptions, etc.). Processing of the exceptions (writing testing examples for starting the exceptions' processing). Integration testing. Testing based on operational profiles. Testing of non-functional requirements (e.g. usability, security, compatibility, accessibility, etc.). Regression testing. Testing tools. Defining the system acceptability. Testing in the domain of DevOps. **15. Specific program architectures** Transaction and analytical software architectures. Critical demands. Design trade off related to goals of the design and development. Data warehouses, OLAP systems, architectures for data mining, ERP systems, distributed systems.
Exercises	Laboratory exercises guide students through the development process of a complex mobile system, focusing the artifacts used and created in each phase, from conceptual modeling through continuous integration, delivery and deployment of final product. During exercises the students will work with the tools covering the full software development lifecycle as well as with technologies of mobile applications development. Agile software development practices are employed as well as industry related tools for source code versioning, project and issues management.
Realization and examination	Classes: lectures and exercises Exam: team project and oral exam
Related courses	University of Goetenborg. - Software Analysis and Design Georgia Tech University / Udacity - Software Architecture & Design University of Alberta / Coursera - Software Design and Architecture Specialization Escuela Politécnica Superior - Software Analysis and Design Project The IEEE Computer Society - Software Design Course University of Sheffield, Object Oriented Programming and Software Design, Software Development for Mobile Devices
Literature	R. Stevens, P. Brook., K. Jackson, S. Arnold: Systems Engineering, Coping with Complexity, Prentice Hall, 1998. M. Fowler with K Scott, UML Distilled: Applying the Standard Object Modelling Language, Addison-Wesley, 1997. S. Bennett, S McRobb R Farmer, Object-Oriented Systems Analysis and Design using UML, McGraw-Hill, 1999. P. Stevens, R Pooley, Using UML - software engineering with objects and components, Addison Wesley, 2000. Sommerville, Software Engineering, 5th edition, Addison-Wesley, 1996. R. S. Pressman, Software Engineering: A Practioner's Approach, 5th edition, McGraw-Hill, 2000 (or the European adaptation by D. Ince).

	T. Gilb, Principles of Software Engineering Management, Addison-Wesley, 1988.