Course title: SOFTWARE DEVELOPMENT

Lecturers	Assoc. Prof. Zlatko Stapić, Ph. D. Full Prof. Vjeran Strahonja, Ph.D. Marko Mijač, Ph.D.
Language of instruction	Croatian and English
Study level	Bachelor
Study programme	Information and Business Systems
Semester	5th (winter)
ECTS	6
Goal	The goal of the Software Development course is to provide students with a thorough overview of the entire field of software product and system engineering and to teach students the methodological development of software products and software development trends. Students will be mentored in the practical and team work assignment of development of complete software product including its features and documentation.
General and specific learning outcomes	
Content	**Lectures** **1. Software development (2)** Trends and requirements for software development. Professional software development and engineering approach to software development. Tools and environments to support development activities. Automated development. **2. Organization of project and software development process (2)** Models and approaches to software development: classic and agile approaches to development. Roles and activities in development processes. Choosing a development approach. Improving the development process. Planning and budgeting of the development process. Agile planning. Risk management. Human resource management. Teamwork. Versioning and quality management. Tools and technologies to support software development project management. **3. Software design practices and specifications (4)** Conceptual modeling. Modeling software product architecture. Architectural decisions. Modeling the structure of a software product. UML modeling. Object-oriented design. Prototyping user interfaces. User experience design. Document the design specification for the structure and behaviour of the software product. Tools to support the design process. **4. Software implementation practices and testing (4)** Integrated development environments. Implementation of object-oriented principles in the selected programming language and development environment. Implementation of UI concepts and experience in selected development environment and UI design tools. Implementation of working with data. Development frameworks. Organization of code. Code versioning and continuous integration. Software product quality assurance. Clean code. Continuous and automated testing and delivery. Tools and technologies for checking and delivering software. **5. Practices for implementing non-functional requirements (2)**

	Classification of non-functional requirements with respect to implementation. Aspects and importance of non-functional requirements in the overall functionality of the software. Redesigning a software product in the context of non-functional requirements. Practices for implementing non-functional requirements. **6. Architectural styles and templates (2)** Architecture design. Decisions in the design process. Views on software architecture. Architectural design styles. Application architectures. Pure architecture. Three-tier architecture. Architecture design templates. **7. Component-based architecture (4)** Component based software engineering. Features of components. Principles of component-based design. Component design guidelines. Carrying out component design. **8. Advanced software engineering (3)** Software frameworks. Software products as services. Service oriented engineering. Software process improvement (SPI). **9. Software Development Trends (3)** Integration of the Internet of Things / Internet of everything development. Aspect oriented programming. Software production lines. Reactive programming in the context of software development. **10. Fundamentals of Software Engineering Economics (4)** Fundamentals of program economics (value chain, costs and benefits; collaborative development, development or purchase, total cost of ownership) Complexity metrics and program cost estimates. Investment decision-making methods (net present value, investment return period)
Exercises	Laboratory exercises are following the content of lectures and practically represent all stages of software development process that are theoretically addressed in lectures. In addition to the practical part of the exercises presented by the teacher, students will develop independent project based on that materials and on the materials available on the e-learning system. Students will apply their own project idea, then plan, design, create and document the product. Finally, the students will defend their work. Students are assigned a mentor who mentors and guides them through the development process.
Realization and examination	Classes: lectures and exercises Exam: preliminary exam and project defense
Related courses	1. Software Engineering, Portland University, 2. (https://www.pdx.edu/computer-science/cs554) 3. The Oregon Master of Software Engineering (OMSE) (http://pdx.smartcatalogiq.com/2019-2020/Bulletin/Courses/OMSE-Software-Engineering) 4. Computing Curricula -- Software Engineering Volume Final Draft of the Software Engineering Education Knowledge (SEEK), April 30, 2003 Edited by Ann E.K. Sobel CCSE Knowledge Area Chair (http://sites.computer.org/ccse/know/FinalDraft.pdf) 5. Software Engineering Institute - SE Curriculum (https://www.sei.cmu.edu/education-outreach/curricula/software-engineering/index.cfm)
Literature	Basic:

	Sommerville I., Software Engineering, 8th edition or newer, Addison Wesley, 2007 or newer Pressman, S. Roger: Software engineering: a practitioner's approach, 7th edition or newer, McGraw-Hill Higher Education, 2010 or newer Optional: Teaching and other course materials available in the learning management system, Authorized sources, web materials, and books on topics that the course addresses, which due to frequent and major changes in technologies and tools are to be defined for each generation of students separately.