Study program / study programs:Software Engineering and Computer Science

Degree level: Master

Course:

Tools and Methods of Software Engineering

Teacher:Đurić O. Dragan,Devedžić B. Vladan,Tomić B. Bojan,Jovanović M. Jelena,Ševarac V. Zoran

Course status: Elective

ECTS points: 6

Prerequisites: Software Development, Software Project Management

Course objective

Mastering contemporary tools and methods of software engineering.

Learning outcomes

Students' ability to use contemporary tools and methods of software engineering in practical projects.

Course structure and content

Theoretical instruction:

MDA methodology and tools. Functional programming methods and tools. Software maintenance tools and methods. Software configuration tools and methods. Software project management tools and methods. Tools and methods for tracking software process (tools for software process modelling, tools for software management, integrated CASE environments). Software quality tools and methods. Heuristic methods based on structure, data, functions, objects and specific domains. Formal methods. Prototype methods. Study example.

Practical instruction:

Labs. Other forms of teaching. Research study work. Work with MDA, EMF, UML, CASE and other software tools in laboratory. Practical project.

Literature/Readings

- Digital learning resources available at the course CD.
- Open source software frameworks and tools, as well as their documentation and useful tutorials are freely available on the Web.

The number of class hours per week				**Other classes:**
Lectures:	**Labs:**	**Workshops:**	**Research study:**	–
2	2	-	–	

Teaching methods Lectures and practical applications			
Evaluation/Grading (maximum 100 points)			
Pre-exam requirements	**Points**	**Final exam**	**Points**
Participation in class		Written exam	
Participation in labs		Project (implementation)	0 – 60
Project (conceptual solution)	0 – 40		

Study program / study programs:Software Engineering and Computer Science

Degree level: Master

Course:

Expert Systems

Teacher:Tomić B. Bojan

Course status: Elective

ECTS points: 6

Prerequisites: none

Course objective

Acquisition of theoretical basis, but also practical skills in design, development and use of expert systems. Developing a critical view of the scope and limits of the practical application of expert systems.

Learning outcomes

Students will develop an understanding of different expert system approaches, methods and techniques. Also, they will be able to develop expert systems by using current technologies in the field.

Course structure and content

Theoretical instruction:

The concept, definition and classification of expert systems.

Expert system architecture.

Methods and techniques for knowledge representation, reasoning and explanation.

Representation of uncertain knowledge.

Software frameworks (frameworks) and tools for expert system development.

Advantages, disadvantages and performance of expert systems.

The use of expert systems and their technologies in specific domains.

Practical instruction:

Exercises, other forms of lectures, research work. Practical work with open source software frameworks, tools and services for expert system development. Development of a practical project.

Literature/Readings

Selected chapters from the following books:

- Durkin, J., Expert Systems - Design and Development, Macmillan Publishing Company, New York, 1994.
- Torsun, I.S., Foundations of Knowledge-Based Systems, Academic Press, NY, 1995.

The number of class hours per week				**Other classes:**
Lectures:	**Labs:**	**Workshops:**	**Research study:**	–
2	2	-	–	

Teaching methods

Lectures and practical applications

Evaluation/Grading (maximum 100 points)

Pre-exam requirements	**Points**	**Final exam**	**Points**
Participation in class		written exam	
Participation in labs		*Project*	0-100

Study program / study programs:Software Engineering and Computer Science

Degree level: II Level- Master Academic Studies

Course:

Elements of dynamical analysis

Teacher:Lazović P. Rade,Mihić R. Olivera

Course status: Elective

ECTS points: 6

Prerequisites:-

Course objective

The course gives an overview of the theory of differential equations and systems of differential equations and introduces elements of dynamical analysis.

Learning outcomes

Students get an introduction into mathematical tools that are used in the analysis of dynamical systems.

Course structure and content

Theoretical instruction:

Ordinary differential equations. Classification. Cauchy problem. Method of successive approximations. Theorems on existence and uniqueness of the solution. Qualitative analysis of ordinary differential equations. Dependence of the solution on initial conditions. Systems of differential equations. Cauchy problem. Method of successive approximations for systems of differential equations. Systems of linear differential equations. Theorems on existence and uniqueness of the solution. Linear independence of solutions. Formula of Liouville. Fundamental system of solutions. Linear systems with constant coefficients. Dynamic systems. General properties. Properties of the solution in the neighborhood of a nonsingular point. Properties of limit trajectories. Orbits and invariant sets. Stability. Lyapunov function. Stability with respect to linear approximation. Examples of application of dynamic systems.

Practical instruction:

Application of software package MATLAB for solving differential equationsand systems of differential equations.

Literature/Readings

1. G. Teschl, Ordinary differential equations and dynamical systems, AMS, 2012

2. M. V. Fedorюk, Obыknovennые differencalьnые uravneniя, Nauka, Moskva, 1980

<table>
<tr><td colspan="5">3. I. G. Petrovskiĭ, Lekcii po teorii differencalьnыh uravneniĭ, URSS, Moskva, 2003

4. D. K. Arrowsmith, C. M. Place, An introduction to Dynamical Systems, AMS, 1992

5. A. Gilat, Uvod u MATLAB sa primerima, Mikro knjiga, Beograd, 2005</td></tr>
<tr><td colspan="4">The number of class hours per week</td><td>Other classes:</td></tr>
<tr><td>Lectures: 2</td><td>Labs: 2</td><td>Workshops:</td><td>Research study:</td><td></td></tr>
<tr><td colspan="5">Teaching methods

Mentoring and computer labs</td></tr>
<tr><td colspan="5">Evaluation/Grading (maximum 100 points)</td></tr>
<tr><td colspan="2">Pre-exam requirements</td><td>Points</td><td>Final exam</td><td>Points</td></tr>
<tr><td colspan="2">Participation in class</td><td></td><td>Written exam</td><td></td></tr>
<tr><td colspan="2">Seminar</td><td>50</td><td>Oral exam</td><td>50</td></tr>
</table>

Study program / study programs: Software Engineering and Computer Science

Degree level: Master

Course:

Intelligent Information Systems

Teacher:Đurić O. Dragan

Course status: Elective

ECTS points: 6

Prerequisites: Intelligent Systems

Course objective

Illustrate how the techniques of artificial intelligence contribute to business information systems improvement. Indicate the directions of expansion of the classical notion of business information systems with the concepts of automated data acquisition and analysis.

Learning outcomes

Students' ability to use various artificial intelligence technologies in information systems development.

Course structure and content

Theoretical instruction:

Introduction. Intelligence in information systems. Typical domains of intelligent information systems application. Important types of IIS. Web mining. Concepst and processes. Web data sources characteristics. Pre-processing of data. Discovering patterns in data on the Web. Interpreting and evaluating patterns. Characteristic Web mining tasks. Selected algorithms for Web mining. Web mining tools. Text mining. Metadata mining. Intelligent Information Systems and machine learning. The concept of machine learning. Tools for applying machine learning in intelligent information systems. Intelligent Information Systems and the Semantic Web. The disadvantages of today's Web from the viewpoint of IIS. Ontology engineering. XML technologies for the Semantic Web. Web resources annotation. Intelligent Web services.

Literature/Readings

1. V. Devedžić (urednik), "Tehnologije inteligentnih sistema", Monografija, Fakultet organizacionih nauka, Beograd, 2004.

2. Devedžić, V.: "Inteligentni informacioni sistemi", digit / Fakultet organizacionih nauka, Beograd,

2000.

3. Digital learning resources available at the course CD.
4. Open source software frameworks and tools, as well as their documentation and useful tutorials are freely available on the Web.

The number of class hours per week				**Other classes:**
Lectures:	**Labs:**	**Workshops:**	**Research study:**	–
2	2	-	–	

Teaching methods

Lectures and practical applications

Evaluation/Grading (maximum 100 points)

Pre-exam requirements	**Points**	**Final exam**	**Points**
Participation in class		Written exam	
Participation in labs		Project (implementation)	0 – 60
Project (conceptual solution)	0 – 40		

Study program / study programs:Software Engineering and Computer Science

Degree level: Graduate studies - master

Course:

Human-Computer Interaction

Teacher:Starčević B. Dušan,Minović V. Miroslav,Milovanović M. Miloš

Course status: elective

ECTS points: 6

Prerequisites: /

Course objective

Training students to define user requirements in domain of human-computer interaction, perform analysis, project, implement and evaluate elements of user interface. All steps are done in accordance with well known and generally accepted development methodologies.

Learning outcomes

Students will acquire necessary knowledge in domain of human-computer interaction, learn to perform analysis, project, implement and evaluate elements of user interface.

Course structure and content

Theoretical instruction:

P-01: Human-computer interaction basics. P-02: Paradigms and principles. P-03: Development process. User models in development process. P-04: Defining user requirements. Social-Technical Models. P-05: Soft systems methodology. Participative development. P-06: Cognitive models. Linguistic models. P-07: Physical and device models. P-08: Assignement analysis. Digital notation and development. P-09: System models: Implementation support. P-10: Evaluation techniques. P-11: Areas of application. Groupware. CSCW. P-12: Multimodal communucation. Speech. Natural user interfaces. P-13: Handwriting recogniton. Computer vision. P-14: Comprehensive computing. Virtual reality. Hypertext. P-15: Multimedia. WWW. Animations. Digital Video. Computer supported learning .

Practical instruction:

V-01: Human-computer interaction basics. V-02: Devices for human-computer interaction. V-03: Principles of user interface. WIMP paradigm examples. V-04: User interface development methodology. V-05: Examples and assigments. V-06: Cognitive systems architecture. V-07: Help systems development. V-08: Decomposition examples (HTA). V-09: Knowledge based analysis example (TAKD). V-10: Analysis based on entity-relationship model (ATOM). V-11: Dialog development examples. V-12: Multimoda communication examples. V-13: Natural user interfaces examples. V-14: Virtual reality examples. V-15: Development of a WWW application with focus on user interface

Literature/Readings

Human-Computer Interaction, Third Edition, Dix, Finlay, Abowd, Beale, Prentice Hall, 2004

Usability Engineering, Jakob Nielsen, Morgan Kaufmann, 1993 Designing the User Interface, Shneiderman, Plaisant, Addison Welsey, 2005				
The number of class hours per week				**Other classes:**
Lectures: 2	**Labs:** 2	**Workshops:**	**Research study:**	
Teaching methods Lectures, labs, practical work, consultations				
Evaluation/Grading (maximum 100 points)				
Pre-exam require*m*ents		**Points**	**Final exam**	**Points**
Project		60	Written exam	40

Study program / study programs:Software Engineering and Computer Science

Degree level: Master

Course:

Combinatorial Optimization and Metaheuristics

Teacher:Stanojević J. Milan,Čangalović M. Mirjana

Course status: Elective

ECTS points: 6

Prerequisites: No

Course objective

To educate students about standard problems of combinatorial optimization and modern metaheuristic methodology for their solving.

Learning outcomes

Students will be capable for individual work on modelling and application of metaheuristics in solving real world combinatorial problems using adequate software.

Course structure and content

Theoretical instruction:

Computation complexity of algorithms and problems. Integer programming. Branch and bound method. Cutting plains method. Optimal paths and trees in graph: shortest path problem, minimal spanning tree problem. Network flows – maximal network flow problem. Traveling salesman problem. Heuristic approach to solving optimization problems. Definition of heuristics. Basic principles of metaheuristic methodologies. Definition of neighborhood. Basic metaheuristic methodologies: simulated annealing, tabu search, variable neighborhood search, genetic algorithms. Examples of application of metaheuristics for solving some of the combinatorial optimization problems: knapsack problem, traveling salesman problem as well as some real world scheduling problems.

Practical instruction:

Application of existing software packages (CONCORDE, GENOCOP) for heuristic solving combinatorial optimization problems.

Literature/Readings

1. Cvetković D., Čangalović M., Dugošija Đ., Kovačević Vujčić V., Simić S., Vuleta J., Combinatorial optimization, mathematical theory and algorithms, Yugoslav Operational Research Society, Belgrade, 1996. (in Serbian)

2. Cook W.J., at al, Combinatorial optimization, John Wiley&Sons, Inc., 1998.

3. Gendreau M., Jean-Yves P. (Ed.), Handbook of Heuristics, Springer, 2010.

4. Günther Z., Roland B., Michael B., Metaheuristic Search Concepts, Springer, 2010.

5. Vujošević M., Optimization methods in engineer management, Faculty of Organizational Sciences, Belgrade, 2012. (in Serbian)				
The number of class hours per week				**Other classes:**
Lectures: 2	**Labs:** 2	**Workshops:**	**Research study:**	
Teaching methods Supervised individual work and/or classical (ex cathedra) with use of computer.				
Evaluation/Grading (maximum 100 points)				
Pre-exam requirements		**Points**	**Final exam**	**Points**
Participation in class		30	Oral exam	70

Study program / study programs:Software Engineering and Computer Science

Degree level: II Level- Master Academic Studies

Course:

Combinatorial Algorithms

Teacher:Čangalović M. Mirjana,Mladenović M. Nenad,Vujčić V. Vera

Course status: Elective

ECTS points: 6

Prerequisites:-

Course objective

Introduction to basic combinatorial objects and review and use of the algorithms for solving related problems. Introduction to basics of graph theory and review of the algorithms for solving the most important problems on graphs.

Learning outcomes

Students will learn the most important combinatorial algorithms and be trained for solving particular combinatorial problems.

Course structure and content

Theoretical instruction:

1. Historical background. Computational complexity of algorithms. 2. Basic combinatorial objects – algorithmic approach. Sorting and searching. Computer presentation of the combinatorial objects. 3. Algorithms for all subsets generation. 4. Algorithms for all combination generation. 5. Algorithms for all permutations generation. 6. Algorithms for all number partitions generation. 7. Algorithms for all set partitions generation. 8. Basic graph theory terms and definitions. 9. Basic graph problems. Computer presentation of graphs. 10. Algorithms for determination of the shortest distances and paths in the graph. 11. Algorithms for all spanning trees generation. 12. Eulerian and Hamiltonian Graphs and Travelling salesman problem – algorithmic approach. 13. Network flows – algorithmic approach. 14. Other combinatorial problems. Looking to the Future.

Practical instruction:

The application of the acquired theoretical knowledge in the specific combinatorial problems by programming and / or existing software packages.

Literature/Readings

1. Jiri Fiala, Jan Kratochvil, Mirka Miller, *Combinatorial Algorithms*, Springer, 2009.

2. Donald Kreher, Douglas Stinson, *Combinatorial Algorithms: Generation, Enumeration and Search*, CRC Press, 1998.

3. Albert Nijenhuis, Herbert S. Wilf, *Combinatorial Algorithms*, Academic Press, 1978.

4. Donald E. Knuth, *The Art of Computer Programming*, Volume 4, Addison-Wesley, 2005.

5. Alan Tucker, *Applied combinatorics*, John Wiley & Sons, 2002.

6. Nicos Christofides, *Graph Theory - an Algorithmic Approach*, Academic Press, 1975.

7. D. Cvetković, M. Čangalović, Đ. Dugošija, V. Kovačević-Vujčić, S. Simić, J. Vuleta, *Kombinatorna optimizacija*, Drustvo operacionih istraživača Jugoslavije, 1996.

The number of class hours per week				**Other classes**
Lectures 2	**Labs** 2	**Workshops**	**Research study**	

Teaching methods

Classroom teaching, Computer.

Evaluation/Grading (maximum 100 points)

Pre-exam requirements	**Points 50**	**Final exam**	**Points 50**
Participation in class	**10**	Oral exam	**50**
Seminar	**40**		

Study program / study programs:Software Engineering and Computer Science

Degree level: Master

Course:

Computer Geometry

Teacher:Stojanović A. Milica,Vučković Đ. Milica

Course status: Elective

ECTS points: 6

Prerequisites: Finished undergraduate studies

Course objective: Actual methods of showing geometry objects. Solving geometry problems by computer.

Learning outcomes: After course, students will be able to create algorithms and problems for solving geometry problems.

Course structure and content

Theoretical instruction: 1. Analytical geometry in the plane and in the space.

2. Graphs: basic facts, usage of graphs in programming.

3. Finding the biggest convex subset in the space.

4. Voronoi diagram in the space.

5. Constructing the convex hull in the space.

6. Finding the nearest neighbors in the space.

7. Polygon. Known examples of triangulation in the plane.

8. Applying Voronoi diagram to the plane problems.

9. Polyhedron and the problem of the triangulation in the space. When the triangulation is possible?

10. Examples of some classes of polyhedra.

11. Algorithms for triangulating polihedra.

12. Triangulation of the set of points in the space.

13. Applying Voronoi diagram to the space problems.

14. Problems in spaces of higher dimension.

15. Seminar work.

Practical instruction:

Creating algorithms in field worked on the theoretical classes.

Literature/Readings

1. Edelsbrunner, H., Algorithms in Combinatorial Geometry, Springer – Verlag, Heidelberg, 1987.
2. Dragan Acketa, Snežana Matić – Kekić, Geometry for informaticars, University in Novi Sad, PMF, Novi Sad 2000. (in Serbian)
3. Trott, Michael, *The Mathematica guide book for graphics*, Springer, 2004.

The number of class hours per week				**Other classes:**
Lectures: 2	**Labs: 2**	**Workshops:**	**Research study:**	

Teaching methods: mentor and/or classical

Evaluation/Grading (maximum 100 points)

Pre-exam requirements	**Points**	**Final exam**	**Points**
Participation in class	15	Written exam	25
Participation in labs		Oral Exam	25
Projects	35		

Study program / study programs:Software Engineering and Computer Science

Degree level: Postgraduate studies (Master academic studies)

Course:

Software construction

Teacher:Lazarević D. Saša

Course status: Optional

ECTS points: 6

Prerequisites: Non

Course objective: Understanding of the principles, rules and methods of construction of software. Getting started with the key issues in software construction. Mastering the methods of construction of software and languages for software design. Using a programming language for the construction of software (coding and testing software). Application of appropriate software tools for constructing of software.

Learning outcomes: Ability of students to develop software using methods, models and tools for software construction and programming in object-imperative language.

Course structure and content (Syllabus):

***Lectures*:**

1. Fundamentals of software construction: Minimizing the complexity and maximizing updatability. Anticipating changes. Techniques to anticipate changes (communication methods, programming languages, platforms, tools). Verification of software.
2. Software construction standards (OMG, IEEE, ISO).
3. management: Models of construction (linear and iterative). Plans for construction.
4. Software construction measurements. Practical Considerations: Design of software construction.
5. Languages for software construction (configuration language, toolkit languages, programming languages). Notation of programming languages (linguistic, formal, visual).
6. Coding (techniques to create the source code, the use of classes, variables, control structures, exception handling, protection code, the organization of source code, documentation of source code).
7. Implementation of XP in the construcion of software.
8. Programming idioms (implementation patterns).
9. Refactoring of source code.
10. Debugging. Testing of source code (unit testing and integration testing).
11. Re-useble software constructions. The quality of software construction. Software integration.

***Labs**:* The order of labs exercises and labs exercise content is fully compliant with lecturing units.

Literature/Readings:

1. I. Sommerville: *Software Engineering*, Addison-Wesley, 2011.
2. S. McConnell: *Code Complete: A Practical Handbook of Software Construction*, Microsoft Press, 2004.
3. B.W. Kernighan and R. Pike: *The Practice of Programming*, Addison-Wesley, 1999.
4. A. Hunt and D. Thomas: *The Pragmatic Programmer*, Addison-Wesley, 2000.
5. M. Fowler: *Refactoring*, Addison-Wesley, 1999.
6. K. Beck, C. Andres: *Extreme Programming Explained: Embrace Change,* 2nd ed., Addison-Wesley, 2004.

The number of class hours per week				**Other classes:**
Lectures: 2	Labs: 2	Workshops: /	Research study: /	/

Teaching methods: *Lectures*: Lectures ex cathedra, and with the use of multimedia resources; specification, implementation, testing; explanation of the case study. *Labs*: case studies, programming.			
Evaluation/Grading (maximum 100 points)			
Pre-exam requirements	**Points**	**Final exam**	**Points**
Participation in class	10	Written exam	40
Project (required)	30	Oral exam	20

Study program / study programs:Software Engineering and Computer Science

Degree level: Master Academic Studies

Course:

Mathematical Programming

Teacher:Vujčić V. Vera,Mladenović M. Nenad,Čangalović M. Mirjana,Mihić R. Olivera

Course status: Elective

ECTS points: 6

Prerequisites: none

Course objective:

The objective is to give an introduction to theory and methods of mathematical programming and the software support for optimizatiion problems.

Learning outcomes

Students learn how to model real-world optimization problems using mathematical programming methodology and how to find optimal solutions using standard software packages.

Course structure and content

Theoretical instruction:

1. Modeling different real-life problems using mathematical programming methodology. 2. Classical optimization. Method of elimination. Method of Lagrange multipliers.3. Onedimensional optimization. Golden section method. Approximation methods. 4. Unconstrained optimization without evaluation of derivatives. 5. Unconstrained optimization of differentiable functions. 6. Convex programming. 7. Nonconvex programming. 8. Nonlinear programming methods. 9. Penalty function methods. 10. Interior point methods for linear and quadratic programming. 12. Global optimization. 13. Software packages for mathematical programming problems. 14-15. Software package GLOB for global optimization.

Practical instruction:

Solving selected mathematical programming problems by standard software.

Literature/Readings

S. Zlobec, J. Petrić, Nonlinear programming, Scientific Publishers, Belgrade, 1989.

2. V. Vujčić, M. Ašić, N. Miličić, Mathematical Programming, Mathematical Institute of the Serbian Academy of Sciences and Arts, Belgrade, 1980.

3. A. Sofer, S. Nash, Linear and Nonlinear Programming, McGraw Hill, 1996.

4. Williams H.P., Model building in Mathematical Programming, John Wiley&Sons, 2003.

The number of class hours per week	**Other classes:**

Lectures: 2	**Labs:** 2	**Workshops:**	**Research study:**	
Teaching methods: Classroom lectures and consultations				
Evaluation/Grading (maximum 100 points)				
Pre-exam requirements		**Points**	**Final exam**	**Points**
Participation in class		30	oral exam	40
Participation in labs		30		

Study program / study programs:Software Engineering and Computer Science

Degree level: Master

Course:

Methods and algorithms of dicrete mathematics in music

Teacher:Manojlović P. Vesna

Course status: elective

ECTS points: 6

Prerequisites: Graduated degree

Course objective: Students will be acquanted with a special application of discrete mathematics in music, using graph theory

Learning outcomes: Students will learn specific terminology and methods from music and discrete mathematics

Course structure and content

Theoretical instruction:

Fundamantals of music theory

Fundamentals of graph theory

Graphs and digraphs

Paths and cycles

Connectedness of graphs and digraphs

Spectra of graphs

Music data bases

Basic notions on data bases

Graph data bases and recognition

Indexing music melodies by graphs

Practical instruction:

Practical work with available data bases

Literature:

D. Cvetković, S. Simić, *Selected chapters from Discrete Mathematics (Odabrana poglavlja iz diskretne matematike)*, 3. edt, Akademska misao, Belgrade 2012
D. Cvetković, *Spectral recognition of graphs*, YUJOR, 22(2012), No. 2, 145-161.
D. Cvetković, **V. Manojolović**, *Spectral recognition of music melodies*, SIM-OP-IS 2013, 269-271

<table>
<tr><td colspan="5">M. F. Demirci, R. H. van Leuken, R. C. Weltkam, Indexing through laplacian spectra, Computer Vision and Image Understanding, 2008. DOI: 10.1016/j.cviu. 2007.09.012
A. Pinto, R. H. van Leuken, M. F. Demirici, F. Niering, P. C. Weltkamp, Indexing music collections through graph spectra, Proc. 8th International Conf. Music Information Retrivial, ISMIR 2007, Wienna, September 23 - 27, 2007, 153-156</td></tr>
<tr><td colspan="4">The number of class hours per week</td><td>Other classes:</td></tr>
<tr><td>Lectures:

2</td><td>Labs:

2</td><td>Workshops:</td><td>Research study:</td><td></td></tr>
<tr><td colspan="5">Teaching methods

Mentoring or classical method (lectures and practical applications)</td></tr>
<tr><td colspan="5">Evaluation/Grading (maximum 100 points)</td></tr>
<tr><td>Pre-exam requirements</td><td>Points</td><td colspan="2">Final exam</td><td>Points</td></tr>
<tr><td>Participation in class</td><td>10</td><td colspan="2">Oral exam</td><td>50</td></tr>
<tr><td>Participation in seminars</td><td>40</td><td colspan="2"></td><td></td></tr>
</table>

Study program / study programs:Software Engineering and Computer Science

Degree level: Master studies

Course:

Advanced software technology

Teacher:Vlajić S. Siniša,Lazarević D. Saša

Course status: Required

ECTS points: 6

Prerequisites: -

Course objective: Gaining knowledge of advanced software technologies used in the development of complex (enterprise) applications. Development of complex applications by using these technologies.

Learning outcomes: Ability of students to design and implement complex applications using advanced software technologies.

Course structure and content

Theoretical instruction:

Overview of advanced software technologies. Realization of multi-tier applications using advanced software technology. The realization of the user interface using modern software technologies. Realization of application logic using modern software technologies. Modern software technology for data access. Modern software technology for the integration of software systems. Software tools for development, testing and evaluation of the quality of a software system. Work with students to develop logical structure of a seminar paper.

Practical instruction:

Developing complex applications using modern software technology. Testing and evaluation of quality of the software applications. The process of developing the study examples.

Literature/Readings

Basic literature:

1. Kim Haase, ***Java(TM) EE 5 Tutorial***, The (3rd Edition) (The Java Series), Addison-Wesley, November 2006
2. Joe Duffy, ***Professional .NET Framework 2.0 (Programmer to Programmer)***, Wrox Press, April 2006

Additional literature:

1. Justin Gehtland, ***Java Enterprise in a Nutshell***, Fourth Edition, O'Reilly, November 2005
2. Ted Neward, ***Effective Enterprise Java***, Addison-Wesley, August 2004
3. Laurence Moroney, ***Java EE and .NET Interoperability : Integration Strategies, Patterns, and***

Best Practices, Prentice Hall, April 2006

The number of class hours per week				**Other classes:**
Lectures: 2	**Labs:** 2	**Workshops:**	**Research study:**	

Teaching methods

• The professor will theoretically explain each of the considered thematic units and by practical examples will explain their use in the development of complex software systems.

• Assistants will elaborate thematic units which professor explained. For each thematic unit assistants will prepare concrete examples that will show and explain to the students in the computer center.

• Students should to do tasks, which will be prepared by assistants.

Evaluation/Grading (maximum 100 points)

Pre-exam requirements	**Points**	**Final exam**	**Points**
Seminar	100	Written exam	

Study program / study programs:Software Engineering and Computer Science

Degree level: Master

Course:

Advanced Software Technologies 2

Teacher:Ševarac V. Zoran,Tomić B. Bojan

Course status: Elective

ECTS points: 6

Prerequisites: none

Course objective

To learn about state-of-the-art software technologies and development practices. To learn several technology standards and frameworks for builing user interfaces and domain models, using coresponding software development tools. To acquire practical skills in software development, and learn about advantages of specific technologies for different application cases

Learning outcomes

Students will learn about building complex software systems using state-of-the-art software technologies and tools.They will acquire practical skills in solving some real-world problem.

Course structure and content

Lectures:

User interface component frameworks. Domain model and data access. Best practice for building application logic and connecting to user interface. Security recommendations.Performance recommendations.Software standards, additional frameworks and tools that supports them.

Labs:

Practical work on software development projects with advanced software frameworks and tools through realistic examples.

Literature/Readings

- *Oracle Java EE7 Tutorial: Java Server Faces Technology, http://docs.oracle.com/javaee/7/tutorial/doc/home.htm*
- *Arun Gupta, Java EE7 Essentials, O'Reilly Media, 2013.*
- *Official Java EE7 specification, http://jcp.org/aboutJava/communityprocess/final/jsr342/index.html*

The *number of class hours* per *week*				**Other classes:**
Lectures: 2	**Labs:** 2	**Workshops:** -	**Research study:** –	–

Teaching methods

Lectures: slides and realistic application studies related to specific software technologies and tools. Learning about basic concepts, features and advantages of specific solutions in software deveopment.

Labs: students work on software development projects under supervision of teaching assistants. Practical work with software tools.

Evaluation/Grading (maximum 100 points)

Pre-exam requirements	**Points**	**Final exam**	**Points**
Participation in class		Written exam	
Participation in labs		Project (implementation)	0 – 100
Project (conceptual solution)			

Study program / study programs:Software Engineering and Computer Science

Degree level: Master

Course:

Numerical Methods in Computing

Teacher:Lazović P. Rade,Đorić S. Dragan

Course status: elective

ECTS points: 6

Prerequisites: none

Course objective

To learn about Floating Point Arithmetic, and IEEE standard 754.

To learn about numerical methods in Linear Algebra and Mathematical Analysis.

Learning outcomes

Student will be introduced to the basic numerical methods, their applications, and state-of -the-art mathematical software packages (MATLAB, MAPLE, MATHEMATICA).

Course structure and content

Theoretical instruction:

Principles of numerical mathematics. Floating point arithmetic. IEEE standard 754. Matrix and vector norms. Matrix factorizations (Cholesky, LU,QR). Eigenvalue problems. Direct methods for solving linear systems. Iterative methods for solving linear systems. Conditioning and stability of linear systems. Polynomial interpolation. Spline interpolation. Numerical methods for solving nonlinear equations and nonlinear systems. Fast Fourier Transformation (FFT).

Practical instruction:

Implementation of numerical methods in MATLAB.

Homework after every chapter.

Exercises and projects.

Literature/Readings

1. C. Gerald, P. Wheatly, Applied Numerical Analysis, California Polytechnic State University, 2004.
2. J. Douglas Faires, R. Burden, Numerical Methods, Thomson Brooks/Cole, 2003.
3. A. Quarteroni, R. Sacco, F. Saleri, Numerical Mathematics, Springer, 2007.
4. A. Gilat, Matlab An Introduction With Applications, John Wiley&Sons, Inc., 2004.

The number of class hours per week				**Other classes:**
Lectures:	**Labs:**	**Workshops:**	**Research study:**	
2	2			

Teaching methods

Evaluation/Grading (maximum 100 points)

Pre-exam requirements	**Points**	**Final exam**	**Points**
Participation in class	10	Written exam	30
Participation in labs	60		

Study program / study programs:Software Engineering and Computer Science

Degree level: II level- Master Academic Studies

Course:

Numerical methods in finances

Teacher:Lazović P. Rade,Đorić S. Dragan,Manojlović P. Vesna

Course status: Elective

ECTS points: 6

Prerequisites: -

Course objective

The course gives an introduction to mathematical models of financial flows and mathematical tools for their analysis

Learning outcomes

Students master application of numerical methods for financial flow analysis.

Course structure and content

Theoretical instruction:

1-2. Introduction. Numerical computations in financial transactions. Relation to numerical methods. Software support. 3-8. Basics of numerical analysis. Errors of approximate values of numbers and functions. Iterative methods for solving systems of linear equations. Direct and iterative methods. Solving systems of nonlinear equations. Approximation of functions. Interpolation. Least-squares approximation. Finite element method for partial differential equations. 9.-12. Mathematical models of finantial flows. Portfolio optimization. Dynamics of price of stock exchange shares. Black-Scholl model. Monte Carlo simulation. Applications of finite element method. 13.-15. Basics of MATLAB. Applications in analysis of mathematical models in finances.

Practical instruction:

Examples of finantial flow models. Implementation of numerical methods in MATLAB. Analysis of mathematical models of financial flows.

Literature/Readings

1. Djurica Jovanov, Numerical Analysis, theory, algorithms, examples, FON, Belgrade, 2005.
2. Rade P. Lazović, Numerical methods, FON, Belgrade, 2013.
3. Rade P. Lazović, Numerical analysis, theory review, examples, problems, FOS, Belgrade, 2009.
4. S. Benninga, Numerical Techniques in Finance, MIT Press, 1989.
5. D. Djorić, Mathematics and MATLAB, Higher School for Electrotechnics, Belgrade, 2003.

6. Paolo Brandimarte: “Numerical Methods in Finance and Economics: A MATLAB-Based Introduction”, John Wiley & Sons, Inc.
7. S. Ross, An Elementary Introduction to Mathematical Finance, Cambridge University Press, 2003.

The number of class hours per week				**Other classes:**
Lectures: 2	**Labs**: 2	**Workshops:**	**Research study:**	

Teaching methods

Classroom teaching and computer labs

Evaluation/Grading (maximum 100 points)

Pre-exam requirements	**Points**	**Final exam**	**Points**
Participation in class	10	Written exam	20
Participation in labs	30	Oral exam	40

Study program / study programs:Software Engineering and Computer Science

Degree level: Master

Course:

Applied artificial intelligence

Teacher:Devedžić B. Vladan,Jovanović M. Jelena,Tomić B. Bojan,Ševarac V. Zoran

Course status: Elective

ECTS points: 6

Prerequisites: none

Course objective

To learn about different state-of-the-art AI (Artificial Intelligence) technologies and techniques.

To examine various application domains of AI technologies, and specific application cases.

To acquire practical skills in the development of intelligent software applications.

Learning outcomes

Students will develop an understanding of different AI technologies and techniques. Equally important, they will acquire practical skills in applying current AI techniques and techniques to develop an intelligent software system that addresses some real-world problem.

Course structure and content

Theoretical instruction:

A comparative analysis of traditional software systems and AI-based software systems.

An overview of the state-of-the-art AI technologies and techniques, as well as their application domains.

Software frameworks and tools for the development of AI-based systems.

The application of the AI technologies on the Web: intelligent Web-based applications.

The application of AI in various domains: education, knowledge management, business, medicine, etc. Case studies.

Practical instruction:

Practical work with publicly available software frameworks and tools for the development of AI-based software systems; project work focused on development of an AI-based system in the domain of student's choice.

Literature/Readings

- Digital learning resources available at the course web site (http://ai.fon.bg.ac.rs/primene-vestacke-inteligencije)
- Open source software frameworks and tools for the development of intelligent systems; all these frameworks and tools, as well as their documentation and useful tutorials are freely available on the

Web.				
The *number of class hours* per *week*				**Other classes:**
Lectures: 2	**Labs:** 2	**Workshops:** -	**Research study:** –	–
Teaching methods Lectures and practical applications				
Evaluation/Grading (maximum 100 points)				
Pre-exam requirements		**Points**	**Final exam**	**Points**
Participation in class			Written exam	
Participation in labs			Project (implementation)	0 – 60
Project (conceptual solution)		0 – 40		

Study program / study programs:Software Engineering and Computer Science

Degree level:Master studies

Course:

Approximate Systems

Teacher:Mihić R. Olivera

Course status:Elective

ECTS points: 6

Prerequisites: /

Course objective:

Argumentation methods for correct, approximate and incorrect inference.

Learning outcomes

Students will learn the techniques of proving (and refuting) based on many-valued, modal, relevant, fuzzy and probabilistic logics..

Course structure and content

Theoretical instruction:

1-3. Many-valued logic as an alternative to classical two-valued logic. Matrix semantics of a finite-valued logic. Hilbert formulation of logical system. Soundness and completeness.

4-7. Infinite-valued logic. Intuitionistic logic as an constructive alternative of mathematics foundations. Kripke possible world semantics. Soundness and completeness.

8-11. Propositional language expansion by modal operators. Normal modal logics, material implication and the possible world semantics.

12-15. Correct, approximate and incorrect inference processes. Statistical syllogism, many-valued, probabilistic and fuzzy logics as a base of founding of the approximate inference definition.

Practical instruction:

Practical classes, other forms of lectures, research work

The topics covered by practical instructions and exercises match the theoretical topics given above

Literature/Readings

1. B. F. Chellas, Modal Logic: An Introduction, Cambridge University Press, Cambridge, 1995.
2. D. van Dalen, Logic and Structure, Springer, Berlin, 1980. (Fifth edition 2013)
3. D. Mundici, Advanced Lukasiewicz Calculus and MV-algebras, Springer, Heidelberg, 2011.
4. Z. Ognjanović, M. Rašković, Z. Marković, *Probability logics*, in Z. Ognjanović (editor), Logic in Computer Science, Zbornik radova 12 (20), Mathematical Institute SANU, Belgrade, 2009, pp. 35-111.
5. G. Priest, An Introduction to Non-Classical Logic, Cambridge University Press, Cambridge, 2008.

The number of class hours per week				**Other classes:**
Lectures: 2	Labs: 2	Other type of classes	Research study:	:
Teaching methods Mentoring or classic teaching				
Evaluation/Grading (maximum 100 points)				
Pre-exam requirements		**Points**	**Final exam**	**Points**
Participation in class		10	Written exam	20
Participation in labs		40	Oral exam	30

Study program / study programs:Software Engineering and Computer Science

Degree level: Master

Course:

Social Network Analysis

Teacher:Jovanović M. Jelena

Course status: Elective

ECTS points: 6

Prerequisites: none

Course objective

To learn about different approaches, methods and techniques that have been developed in the field of Social Network Analysis (SNA). To examine typical application domains and specific application cases, and thus develop a good understanding of pros and cons of individual SNA methods and techniques.

To acquire practical skills in the analysis of network data, using publicly available SNA software tools.

Learning outcomes

Students will develop an understanding of different SNA approaches, methods and techniques. They will also get an insight into the potentials and relevancy of these methods and techniques in different application domains. Last, but not the least important, they will acquire practical skills in applying SNA methods and techniques to real-world problems.

Course structure and content

Theoretical instruction:

Basic concepts: graph-based data representation (nodes, edges, adjacency matrix, etc.); network features (degree, paths, diameter, connected components, etc.).

Random network models: Erdos-Renyi model and Barabasi-Albert model

Centrality measures (degree centrality, betweeness centrality, eigen vector centrality, etc)

Community detection.

Small world phenomenon and the related network models.

Models of strategic network formation.

Diffusion in a network: the impact of the network structure on the interaction of network members; opinion formation in a network; the diffusion of information and innovation through a network.

Networked learning.

Practical instruction:

Practical work with publicly available software tools for SNA (e.g., Gephi, R) and real-world network

datasets; covering the topics encompased by the theoretical instruction.

Literature/Readings

Selected chapters from the following books:

- M.O. Jackson. 2010. *Social and Economic Networks*. Princeton University Press, USA
- D. Easley and J. Kleinberg. 2010. *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*. Cambridge University Press, New York, NY, USA.

The number of class hours per week				**Other classes:**
Lectures:	**Labs:**	**Workshops:**	**Research study:**	–
2	2	-	–	

Teaching methods

Lectures and practical applications

Evaluation/Grading (maximum 100 points)

Pre-exam requirements	**Points**	**Final exam**	**Points**
Participation in class		written exam	0 – 60
Participation in labs			
Project	0 – 40		

Study program / study programs:Software Engineering and Computer Science

Degree level: Master studies

Course:

Software requirements

Teacher:Vlajić S. Siniša

Course status: Election

ECTS points: 6

Prerequisites: Software design

Course objective: Introduction to the process of requirements gathering. Mastering the techniques of the requirements gathering and forms of the specifications and validation of the requirements.

Learning outcomes: Students need to get through their own case study learn the process of requirements gathering, specification and validation requirements.

Course structure and content

Theoretical instruction:

Basics of software requirements: Definitions software requirements. The main types of requirements. Quantifying requirements. The difference between system and software requirements. The process of requirements gathering: Define the process. Model processes. Management of the processes. The quality of the process. Getting requirements: Sources of the software requirements. Collection and organization of the requirements. Techniques of requirements gathering. Requirements analysis: The limits of a software system. Interaction with the environment. Define system requirements. Classification requirements. Conceptual modeling. Requirements specification: the forms of the requirements specification, verification, validation of the requirements. Validation of the requirements: Rating requirements. Verification requirements. Prototyping. Validation of the model. Tests. Practical consideration: iterative nature of the process of gathering requirements. Change management requires. Attribute of the requirements. Routing of the requirements. Measurement of the requirements.

Practical instruction:

Quantifying requirements, requirements gathering techniques, classification of the requirements, evaluation of the requirements, assessment of the requirements, prototyping, model validation, testing, attributes of the requirements and measurement of the requirements.

Literature/Readings

Basic literature:

1. Pohl K., Requirements Engineering Fundamentals, Principles, and Techniques, 2010
2. Hull E., Jackson K.,Dick J., Requirements Engineering, Springer, 2011.
3. Sommerville, I.,: Software Engineering, 8th., Addison-Wesley, 2006.
4. Klaus P., Rupp ., Requirements Engineering Fundamentals: A Study Guide for the Certified Professional for Requirements Engineering Exam - Foundation Level - IREB compliant, Rocky Nook, 2011

5. Savić Dušan, Siniša Vlajić: Software requirements,book in preparation, 2011..

Additional literature:

1. R.R. You: Effective Requirements Practices,Addison-Wesley, 2001.
2. G. Kotonya and I. Sommerville: Requirements Engineering: Processes and Techniques, John Wiley & Sons, 2000.
3. R.H. Thayer and M. Dorfman, eds.: Software Requirements Engineering, IEEE Computer Society Press, 1997, pp. 176-205, 389-404.
4. S. Robertson and J. Robertson: Mastering the Requirements Process, Addison-Wesley, 1999.

The *number of class hours* per *week*				**Other classes:**
Lectures: 2	**Labs:** 2	**Workshops:**	**Research study:**	

Teaching methods

• The professor will theoretically explain each of the considered thematic units and by practical examples will explain their use in the development of complex software systems.

• Assistants will elaborate thematic units which professor explained. For each thematic unit assistants will prepare concrete examples that will show and explain to the students in the computer center.

• Students should to do tasks, which will be prepared by assistants.

Evaluation/Grading (maximum 100 points)

Pre-exam requirements	**Points**	**Final exam**	**Points**
Seminar	100		

Study program / study programs:Software Engineering and Computer Science

Degree level: Master studies

Course:

Software process

Teacher:Vlajić S. Siniša

Course status: Election

ECTS points: 6

Prerequisites: -

Course objective: Gaining knowledge about the software process which is defined by its models, methods, strategies and phases. Mastering the models and methods of the process assessment.

Learning outcomes: Ability of students to develop a software system in accordance with the standard models, methods and strategies of the software process.

Course structure and content

Theoretical instruction:

Basics of the software development process (software process). Software system. Models of the business system (structural system analysis, process analysis, ...). Models of the software process (Iterative-incremental, Model waterfalls, ..., spiral model). Methods of software process (Larman method, the Unified software development process, ..., Scrum, Extreme Programming). Strategy of the software process (a process driven by use cases, a process driven by models ... a process driven by tests). Phases of the software process. Infrastructure and process management software. Adaptation and process automation. Evaluation of the software process and software product.

Practical instruction:

Defining the business system, the business system modeling, iterative-incremental software process model, a process driven by use cases, a process driven by models and a process driven by tests. Process automation and evaluation of the software process and software product.

Literature/Readings

Basic literature:

Siniša Vlajić: *Software process*, Book in preparation, 2011.

Additional literature:

1. Object Management Group: Software Process Engineering Metamodel Specification, 2002,

http://www.omg.org/docs/formal/02-11-14.pdf.

2. S.L. Pfleeger, Software Engineering: Theory and Practice, second ed., Prentice Hall, 2001.

3. R.S. Pressman, Software Engineering: A Practitioner's Approach, sixth ed., McGraw-Hill, 2004.

4. K.H. Bennett and V.T.Rajlich, Software Maintenance and Evolution: A Roadmap, The Future of Software Engineering, A. Finklestein, ed., ACM Press, 2000.

5. K.H. Bennett, "Software Maintenance: A Tutorial in software Engineering, M. Dorfman and R. Thayer, eds., IEEE Computer Society Press, 2000.

The number of class hours per week				**Other classes:**
Lectures: 2	**Labs:** 2	**Workshops:**	**Research study:**	

Teaching methods

• The professor will theoretically explain each of the considered thematic units and by practical examples will explain their use in the development of complex software systems.

• Assistants will elaborate thematic units which professor explained. For each thematic unit assistants will prepare concrete examples that will show and explain to the students in the computer center.

• Students should to do tasks, which will be prepared by assistants.

Evaluation/Grading (maximum 100 points)

Pre-exam requirements	**Points**	**Final exam**	**Points**
Seminar	30	Exam on the computers	20
		Oral exam	50

Study program / study programs:Software Engineering and Computer Science

Degree level: Master

Course:

Theory of the Algorithms

Teacher:Stojanović A. Milica,Manojlović P. Vesna

Course status: Required/Elective

ECTS points: 6

Prerequisites: Finished undergraduate studies

Course objective: Presentation of the basic elements of the numerical complexity and analysis of the algorithms. Teaching students to make algorithms in different fields (graph theory, algebra, geometry, sequences, set theory)

Learning outcomes: After course, students will be able to create algorithms and to determine their numerical complexity.

Course structure and content

Theoretical instruction:

1. Time and space complexity of an algorithm and a problem. 2. Deterministic and nondeterministic Turing machine. 3. NP class of problems. NP completeness and NP hard problems. 4. Construction of algorithms by the induction, examples. 5. Strengthening the inductive hypothesis; proving correctness of the algorithm. 6. Algorithms on the graphs: detour in graph; the shortest paths. 7. Problem of the matching in the graph; transportation network; Hamiltonian paths. 8. Geometrics algorithms: problems with polygon; convex hull. 9. Algebraic algorithms: problems with polynomials. 10. Problems with matrices. 11. Algorithms over sequences and sets. 12. Some of the algorithms in cryptography. 13. Parallel algorithms; algorithms for computer networks. 14. Seminar work.

Practical instruction:

Creating algorithms in field which were studied theoretically and analysis of their complexity.

Literature/Readings

1. M. Živković: Algorithms, Math. Faculty, Belgrade, 2000. (in Serbian)
2. Z. Ognjanović, N. Krdžavac: Introduction into theoretical computer science, FON, Belgrade, 2004. (in Serbian)
3. Leung Joseph, ed.: Handbook of scheduling: algorithms, models, performance analysis, Boca Raton [etc.]: Chapman and Hall/CRC, 2004.

The number of class hours per week				**Other classes:**
Lectures: 2	**Labs: 2**	**Workshops:**	**Research study:**	

Teaching methods: mentor and/or classical

Evaluation/Grading (maximum 100 points)			
Pre-exam requirements	**Points**	**Final exam**	**Points**
Participation in class	15	written exam	25
Participation in labs		oral exam	25
Project	35		

Study program / study programs:Software Engineering and Computer Science

Degree level: Master

Course:

Graph theory

Teacher:Čangalović M. Mirjana,Manojlović P. Vesna

Course status: elective

ECTS points: 6

Prerequisites: Undergraduate studies

Course objective

The aim is to introduce students to basic notions of the graph theory, especially notions related to trees, as well as to basic concepts of the theory of graph spectra and its applications in the computer science.

Learning outcomes

Introducing to some important applications of the theory of graph spectra in the computer science, such as applications within complex networks and Internet, Internet search, antivirus protection, statistical data bases, social networks and quantum computers.

Course structure and content

Basic definition of the graph.
Graph representations: The adjacency matrix, the incidence matrix of vertices and edges, the distance matrix. Euler and Hamilton paths in the graph.
Trees: Basic definitions, rooted trees, binary trees and their applications to the computer science. Some optimization problems on graphs: shortest path problem, minimal spanning tree problem, travelling salesman problem.
Spectra of graphs and its applications: Laplacian matrix of the graph.
Eigenvalues and eigenvectors of graphs. Basic characteristics of the spectrum of a graph. Some applications to the computer science. Antivirus protection, Internet search, sportsmen ranking, pattern recognition.

Literature

D. Cvetković, M. Čangalović, Dj. Dugošija, V. Kovačević Vujčić, S. Simić, J. Vuleta, *Kombinatorna optimizacija (Combinatorial Optimisation)*, Dopis, Belgrade 1996

J.A. Anderson, *Diskretna matematika sa kombinatorikom*, Računarski fakultet, 2005

M. Čangalović, **V. Manojlović**, V. Baltić, *Diskretne matematičke strukture*, FON, 2009

D. Cvetković, P. Rowlinson, S. Simić, *An Introduction to the Theory of Graph Spectra*, Cambrige University Press, 2009

Selected Topics on Applications of Graph Spectra, *Compendium 14 (Zbornik radova)*, Institute for Mathematics – the Serbian Academy of Science & Art, Belgrade 2011

D. Cvetković, S. Simić, *Graph Spectra in Computer Sciences, Linear Algebra and Applications*, Belgrade 2011

The number of class hours per week				**Other classes:**
Lectures: 2	**Labs:** 2	**Workshops:**	**Research study:**	
Teaching methods Classical lectures illustrated by corresponding software implementations				
Evaluation/Grading (maximum 100 points)				
Pre-exam requirements	**Points**	**Final exam**	**Points**	
Participation in class	**10**			
Participation in labs	**40**	Oral exam	**50**	

Study program / study programs:Software Engineering and Computer Science

Degree level: Postgraduate studies (Master academic studies)

Course:

Testing and software quality

Teacher:Lazarević D. Saša

Course status: Optional

ECTS points: 6

Prerequisites: Non

Course objective: **Part I**: Understanding of the principles, rules and methods of software testing. Introduction to techniques of software testing. Mastering the process of software testing. Utilising the available development environment for testing software. The development of software-driven testing. ◆ **Part II**: Understanding the principles, rules and methods of software quality. Specifying the models and the features of software quality. Understanding and mastering the process of quality management software. Metrics. Optimization and performance tuning. Application of appropriate software tools for managing software quality.

Learning outcomes: Competence of students as to test the software using methods, models and tools for software testing, as well as to optimize the software.

Course structure and content (Syllabus):

***Lectures*: Part I**: Fundamentals of Software Testing: The terminology of software testing. Key testing questions (dynamic, finality, selectivity, expectancy). Link testing with other activities of software development. Testing levels: test's subject. Test Purposes (qualification testing, installation testing, alpha and beta testing, correctness testing, reliability testing and evaluation, regression testing, performance testing, etc). Testing techniques: Techniques based on the experience of the tester. Techniques based on the specification of the program. Techniques based on the program code. Techniques based on errors of programs. Techniques based on the use of the program. Techniques associated with the nature of the application. Combining techniques. The measurements related to the test: Evaluation of the program to be tested. Evaluation of the tests. Testing process: Process control testing. Test documentation. Test models. Testing activities. ◆ **Part II**: Fundamentals of Software Quality: Ethics and the culture of software engineering. Value and cost of quality. Models and quality characteristics (quality of the software process, the quality of a software product). Quality improvement. Process Quality Management Software: Security software quality. Verification and validation. Review and monitoring of software quality (management review, technical review, inspection anomalies, evaluation of software products, testing software product). Practical Considerations: Requirements of software quality (impact factors, dependence, levels of integrity software). Properties of the defect (error, fault, failure, mistake). Techniques of software quality (static techniques, oriented towards people, analytical techniques, dynamic techniques, testing). Measuring software quality (statistical measure, trend analysis and prediction). Metrics. Performance tuning software.

***Labs**:* The order of labs exercises and labs exercise content is fully compliant with lecturing units.

Literature/Readings:

1. K. Beck: *Test-Driven Development by Example*, Addison-Wesley, 2002.
2. P. C. Jorgensen: *Software Testing: A Craftsman's Approach*, 2nd ed., CRC Press, 2004.
3. C. Kaner, J. Bach, and B. Pettichord: *Lessons Learned in Software Testing*, Wiley Comp. Publishing, 2001.
4. S. L. Pfleeger: *Software Engineering: Theory and Practice*, 2nd ed., Prentice Hall, 2001.

5. J. W. Horch: *Practical Guide to Software Quality Management*, Artech House Publishers, 2003.
6. S.H. Kan: *Metrics and Models in Software Quality Engineering*, 2nd ed., Addison-Wesley, 2002.
7. S. McConnell: *Code Complete: A Practical Handbook of Software Construction*, Microsoft Press, 2004.
8. I. Sommerville: *Software Engineering*, 7th ed., Addison-Wesley, 2005.

The number of class hours per week				**Other classes:**
Lectures: 2	Labs: 2	Workshops: /	Research study: /	/

Teaching methods: *Lectures*: Lectures ex cathedra, and with the use of multimedia resources; specification, implementation, testing; explanation of the case study. *Labs*: case studies, programming.

Evaluation/Grading (maximum 100 points)

Pre-exam requirements	**Points**	**Final exam**	**Points**
Participation in class	10	Written exam	40
Project (required)	30	Oral exam	20

Study program / study programs:Software Engineering and Computer Science

Degree level: Graduate studies (Master)

Course:

Security Techniques in Computer Networks

Teacher:Simić B. Dejan,Starčević B. Dušan

Course status: Elective

ECTS points: 6

Prerequisites: /

Course objective The course objective is to transfer knowledge to students about possible threats, attacks, and safeguards that are relevant to Internet environment, and Web services, the basic principles of protection techniques and mechanisms for the protection of information systems and computer networks, various methodological approaches to the design and implementation of protection.

Learning outcomes Students will gain the necessary knowledge in the field of computer networks security on concrete examples.

Course structure and content

Theoretical instruction:

L-01: Introduction to Network Security. L-02: Basic Concepts of Network Security. L-03: Security Models. L-04: Access Control Mechanisms. L-05: Introduction to Cryptography. L-06: Applied Cryptography. L-07: Digital Signature. L-08: Digital Certificates. L-09: SSL/TLS protocol. L-10: IPsec. L-11: Intrusion Detection and Prevention Systems. L-12: Network Security and Wireless Security. L-13: Protecting Applications in Computer Networks. L-14: Electronic Payment Systems Security. L-15: Review of previous lectures and preparing for the exam.

Practical instruction: *Exercises, Other forms of lectures, Research work:*

E-01: Basic Terms in Network Security. E-02: Risk Management Methods. Social Engineering Methods. E-03: Protocols for Network Security. E-04: Nessus E-05: Examples of malicious software (malware) in computer networks. E-06: Linux operating system protection. E-07: Windows operating system protection. E-08: Kerberos. E-09: Examples of Applied Cryptography in Computer Networks. E-10: Steganography. Web Security. E-11: Authentication Methods. E-12: Appying Smart Cards in Computer Networks. E-13: Appying PKI. E-14: Applying Firewalls. E-15: Review of previous excercises and preparation for the exam.

Literature/Readings

1. Lectures in e-form, FON, 2013.

2. Jim Curose, Keith Ross, *Computer Networking: A Top Down Approach*, 6th edition, Addison-Wesley, 2012.

3. Stallings W., *Network Security Essentials: Applications and Standards*, Pearson Education Limited, 2013.

4. Randy Weaver, *Guide To Network Defense and Countermeasures*, 3rd edition, 2013.
5. Emmett Dulaney, *ComTIA Security+ Deluxe Study Guide*, Sybex, 2009.

The number of class hours per week				**Other classes:**
Lectures: 2	**Labs:** 2	**Workshops:**	**Research study:**	

Teaching methods

Lectures, Exercises, Practical Work, Consultation.

Evaluation/Grading (maximum 100 points)

Pre-exam requirements	**Points**	**Final exam**	**Points**
Participation in class	30	Written exam	30
Participation in labs	40		

Study program / study programs:Software Engineering and Computer Science

Degree level: Master

Course:

Software Configuration Management

Teacher:Đurić O. Dragan,Devedžić B. Vladan

Course status: Elective

ECTS points: 6

Prerequisites: Software Development

Course objective

Mastering models, methods and techniques of software configuration management.

Learning outcomes

Familiarizing with SCM process. Understanding how software configuration is identified and controlled. Mastering the process of software development and shipping.

Course structure and content

Managing the SCM process: Organizational context of SCM. Contraints and SCM management. Planning SCM. SCM maintenance (measuring and examining). Software configuration identification: Identifies appearances that are to be controlled (software configuration, configuration of software instances, configuration of the links between software instances, software versions, acquiring instances). Software library. Software configuration control: change management during the software lifecycle. Requirement, evaluation and confirmation of software changes. Software change implementation. Deviations and change release. Determining software configuration status: Status. Reports. Sovtware configuration guidance: Software function configuration management. Physical software configuration management. Base points configuration management. Implementation and deployment management: Software implementation. Software deployment.

Literature/Readings

1. R.S. Pressman: Software Engineering: A Practitioner's Approach, Sixth ed, McGraw-Hill, 2004.

2. W. Royce: Software Project Management, A United Framework, Addison-Wesley, 1998.

3. I. Sommerville, Software Engineering, seventh ed., Addison-Wesley, 2005.

4. IEEE Std 828-1998: IEEE Standard for Software Configuration Management Plans, IEEE, 1998.

5. Anne Mette Jonassen Hass: Configuration Management Principles and Practice,Addison-Wesley, 1999.

6. Brad Appleton: Software Configuration

The number of class hours per week				**Other classes:**
Lectures:	**Labs:**	**Workshops:**	**Research study:**	–
2	2	-	–	

Teaching methods

Lectures and practical applications

Evaluation/Grading (maximum 100 points)

Pre-exam requirements	**Points**	**Final exam**	**Points**
Participation in class		Written exam	
Participation in labs		Project (implementation)	0 – 60
Project (conceptual solution)	0 – 40		

Study program / study programs:Software Engineering and Computer Science

Degree level: Master

Course:

Practice Specification

Teacher: All teachers involved in the study program

Course status: Mandatory

ECTS points: 4

Prerequisites: /

Course objective

Training students to do independent research and professional work in identifying and solving specific tasks in the program of study, in real conditions of practice and / or research laboratories and centers.

Learning outcomes

Gaining experience and mastery of skills in the use of deepening and enriching the acquired theoretical and practical knowledge for the purpose of identifying and resolving specific issues and tasks that occur in the real system.

Course structure and content

Elements of the project task; Defining the objectives and tasks of the research; Identification and description of the basic problems through the development of key thesis; The basic methods, techniques and tools for the project professional practice - selection of appropriate methods TOR and predicted empirical research; Basic elements of the presentation of research results - the principles of successful presentations and various forms and characteristics of individual forms, such as the content of written documents, oral, electronic presentations; Defining a specific project task of professional practice for each student - goals and tasks, duties and responsibilities of the student organization (if it is implemented in a particular organization), mode, form and content of the final report, and etc.

Literature/Readings

The *number of class hours* per *week*				
Lectures:	**Labs:**	**Workshops:**	**Research study:**	**Other classes:**
			20	

Teaching methods

The application of different methods of research, consultations (individual and group). The use of different teaching methods with practical work.

Evaluation/Grading (maximum 100 points)

Pre-exam requirements	**Points**	**Final exam**	**Points**

Seminar	50	Written exam	50

Study program / study programs: Software Engineering and Computer Science

Degree level: Master

Course:

Research proposal

Teacher: All teachers involved in the study program

Course status: Mandatory

ECTS points: 8

Prerequisites: /

Course objective

The main objective is to prepare students for Degree - Master of work, so he is the first phase of development of master work. With the help of mentors, students will be prepared that, with the conquest of the necessary methods and with the use of basic acquired during their studies, scientific-technical and professional application of knowledge, solve a specific problem within the selected areas. As part of these preparations student studying the broader context of the problem, its structure and complexity.

Based on literature student meets with the existing approaches to solving similar tasks and good practice. Based on the conducted comparative analysis of available solutions student brings a proposal of its own approach to solving the complex problems. The aim of the activities of students in this part of the research is to gain the necessary experience through solving complex problems and tasks and identifying opportunities for the application of previously acquired knowledge into practice.

Learning outcomes

Engineer should improve their previous titles acquired those skills and knowledge which enables him to solve the most complex problems. In addition to the knowledge and skills acquired in undergraduate studies, students are trained for research work. Acquire the necessary knowledge in specific scientific fields, methods of scientific research and skills (oral presentation, group communication, etc.). Because creative approach to the interpretation of other people's knowledge and experience can exercise and less scientific contributions. In this way gain a better performance on the market work, and acquired competencies enable them to find employment in research and development centers and institutes, or in companies that are committed to improving their own work and open to new approaches and solutions in the areas of organization and management. In the access student work defines the topic, purpose, research methods, literature you will use.

Course structure and content

The content of the work depends on the particular rešavanog problems and is aligned with the objectives of the case. The work includes the object and purpose of the research, initial hypotheses, research methods, the contribution of access and conclusions.

Literature/Readings

The *number of class hours* per *week*	**Other**

Lectures:	Labs:	Workshops:	Research study:	classes:
			20	

Teaching methods

After discussions with the supervisor about topics of the future specialist labor, student, with the approval of the selected mentors and task-specific, starts making the access operation. During the preparation of this paper, mentor conduct regular consultations to learn about the progress of the student, critically evaluate current work and provides additional guidance in the form of student guidance or reference to a particular literature.

Evaluation/Grading (maximum 100 points)

Pre-exam requirements	Points	Final exam	Points
Creation paper specification	50	Defense graduate paper specification	50

Study program / study programs:Software Engineering and Computer Science

Degree level: Master

Course:

Graduate paper specification

Teacher: All teachers involved in the study program

Course status: Mandatory

ECTS points: 18

Prerequisites: /

Course objective

Engineer of organizational sciences should demonstrate an increased ability to research in the case of new or unfamiliar problems in this area, linking the acquired knowledge and skills in solving complex problems, and the ability to follow and adopt papers and research results.

Learning outcomes

Graduate engineers - masters improve their previous knowledge acquired those skills and knowledge that they provide better performance on the market work, and acquired competencies enable them to find employment in research and development centers and institutes, enterprises or their own organizations. Students gain specialization in the above sub-group can independently or in a team to solve the most complex problems, because they deepen previously acquired academic skills and knowledge, understanding and skills. Are trained to solve complex problems. They independently investigate, process the data obtained in the research, draw conclusions, write and defend the results.

Course structure and content

By creating and defending the master's thesis students are usavšavaju in the scientific field that is the subject of their master academic studies and acquire a graduate engineer in the field of master academic studies. Engineer - master has deepened academic theoretical and practical knowledge and skills in the chosen specific scientific field, knows in academia and beyond the accepted methodology for solving complex problems and is able to be independent and creative application in solving the problems that will occur in practice.

Literature/Readings

The *number of class hours* per *week*				**Other classes:**
Lectures:	**Labs:**	**Workshops:**	**Research study:**	

Teaching methods

After accepting the diploma master work of a candidate under the supervision of a mentor approach to designing work. Creating work should be carried out in accordance and in the implementation plan exposed in the application work. Candidate in the laboratory and / or field work independently on the practical aspects of the problems solved. In consultation with the supervisor if necessary checks the work plan, in terms of the elements it contains, or the dynamics of additional sources.

Evaluation/Grading (maximum 100 points)			
Pre-exam requirements	**Points**	**Final exam**	**Points**
Creation graduate paper specification	50	Defense graduate paper specification	50