UNIVERSITETI I EVROPËS JUGLINDORE
УНИВЕРЗИТЕТ НА ЈУГОИСТОЧНА ЕВРОПА
SOUTH EAST EUROPEAN UNIVERSITY

Study program	Software Engineering
Faculty	Contemporary Sciences and Technologies
Study Cycle	Second Cycle (Postgraduate)
ECTS	120
Accreditation date	30.11.-0001

Description of the program

The Masters programme in Software Engineering is designed to provide students with opportunities to advance their knowledge in specific Computer Science fields, to direct themselves to the research field, as well as to learn to apply the acquired theoretical and technical skills both in the industry and the business sector.

The programme nurtures creative thinking and innovative approach to solving problems through solid theoretical background and new technologies.

The plan-programme for Masters Studies in Software Engineering is based upon the adopted Bologna model of organizing the studies with a duration of four semesters, also suitable for the students completing undergraduate studies in three-year programmes.

Career

The programme will enable students with the necessary knowledge and skills to contribute in all the aspects of the software development process, including planning, collaboration, specifications, design, development, delivery and maintenance of software products. In addition, students will also acquire general skills, such as analytical and critical thinking, team work and work in multicultural environments, planning and organizing etc.

After finishing this programme, graduates will have career opportunities in a variety of industries, mainly fulfilling needs for computer systems design, such as software developer, software tester, manager of software projects and processes.

The last semester of studies includes master thesis writing, enabling program graduates to continue their studies towards a doctoral degree in computer sciences.

Learning outcomes

Knowledge and understanding

- Advanced knowledge and understanding in the field of software and application development, including:
- Software development, needs analysis, design, coding, testing;
- Programming languages, their analysis and use in the development of various software solutions, software system analysis, development of Internet applications and security, networking;
- Knowledge of advanced concepts in computer science;
- Managing large software projects.

Applying knowledge and understanding

- Can apply, use, develop and deploy advanced software systems.
- Can offer and apply different methods and methodologies of software development for delivery of major IT solutions.
- Can use various tools for software development and program them using shell, scripts and compiled programs in standalone or web environments.
- In an original, critical and creative way can participate in the process of solving problems in new, unseen or unknown environments for software development.
- Is able to organize software systems in order to solve various social, economic and/or technological issues.
- Is able to participate in research projects as a base for further academic development.
- Demonstrates expertise in addressing real problems in the field of software development and project management.
- Can develop and apply original and creative ideas.

Making judgement

- Can in an adequate way collect, analyze and evaluate data using modern tools and systems for certain social, economic and/or organizational issues.
- Is able to adequately assess the required deadlines, resources and risks in the planning, development, deployment and maintenance of software, using appropriate tools.
- Can argue and explain ideas and concepts.
- Can test, assess and appropriately decide on various possible IT solutions.

Communication skills

- Can in a clear and unambiguous way communicate knowledge, data and study results to team members, customers, managers and other stakeholders in software development.
- Can adequately adjust the style and form of expression when addressing unskilled audience.
- Can initiate, lead and take responsibility for the work of a group of people.
- Is able to undertake preparations for research and contribute in the field of software development.

Learning skills

- Can follow new developments in the field of software and application development, learn new technologies and implement them.
- Can identify their needs and directions of personal and autonomous development.

List of courses

Semester 1

- [6.0 ECTS] **Research Methodology**
- [6.0 ECTS] **Advanced Data Structures and Algorithms**
- [6.0 ECTS] **Software Processes and Management**
- [6.0 ECTS] **Seminar Paper**
- [6.0 ECTS] **Free elective course 1**

Semester 2

- [6.0 ECTS] **Software Analysis and Design**
- [6.0 ECTS] **Programming Paradigms**
- [6.0 ECTS] **Distributed Systems**
- [6.0 ECTS] **Laboratory Course**
- [6.0 ECTS] **Free elective course 2**

Semester 3

- [6.0 ECTS] **Software Testing**

- [6.0 ECTS] **Team Project**
- [6.0 ECTS] **Automata Theory and Formal Languages**
- [6.0 ECTS] **Operating System Security**
- [6.0 ECTS] **Elective course 1**

Semester 4

- [30.0 ECTS] **Master Thesis**

Description of courses

Core courses

- **Research Methodology**
  The goal of this course is to enable students with knowledge and understanding of different scientific theories and methodologies that are applied in the field of software engineering. After completing the course, students will be able to: * thoroughly explain and understand the importance of the basic scientific concepts; * effectively search relevant information and literature; * identify, describe and formulate scientific problems; * make a careful choice of alternative research approaches; * thoroughly describe, compare and explain the advantages and disadvantages of various scientific methods for collecting quantitative and qualitative data; * apply basic scientific methods during analysis of quantitative and qualitative data; * understand different frameworks for theory building; * assess and review scientific publications.

- **Advanced Data Structures and Algorithms**
  This course is based on previous knowledge of algorithms and data structures. The course goal is to acknowledge students with advanced efficient algorithms and appropriate data structures used for data presentation, search and optimization. The course also reviews the theory of algorithm complexity and its practical determination in order to be able to compare different algorithms. During the course, students will become familiar with several known algorithms, especially for search and optimization in complex nonlinear structures, such as trees and graphs.

- **Software Processes and Management**
  The goal is to provide students with a deep, critical and systematic understanding of the principles and techniques for manageing the design of effective software applications. The outcome of this course is to present to students a methodology of understanding the management of the development of software solutions. The objectives of the course are: profiles and types of software projects, understanding of creating the software project structure, managerial approach to solving problems and tools, evaluation and management of risk, monitoring and control, maintaining quality, techniques for managing agile software development, etc.

- **Seminar Paper**
  This course covers various aspects of management of the development of software systems and work in groups and managing the development techniques of agile software development. The students are given the opportunity to prepare paper work under supervision, to highlight their knowledge and skills by using software tools for tracking and monitoring of projects, software tools for teamwork, organizational decision-making potential of an effective system of management information, organizational learning, a source of competitive advantage; organization in a global arena.

- **Software Analysis and Design**
  Thorough review of the software design. Continuation of the study of design patterns, design and architecture frameworks. Review of current middleware architectures. Design of distributed systems using middleware. Component-based design. Theory of measure and use of metrics in the design. Features of a good design: performance, reliability, security, reusability, reliability, etc. Measuring internal qualities and complexity of the software. Evaluation and evolution of the design. Basics of software evolution, reengineering and reverse engineering.

- **Programming Paradigms**
  The course provides an overview of the principles on programming languages design and topics related to their implementation. The topics include comparison of the most important programming paradigms: declarative programming, functional, procedural, logical, relational, object-oriented and other programming paradigms. Furthermore, data types, methods of determining language semantics and syntax, control structures, data connecting

and other related topics are covered. An introduction to alternative programming languages, such as functional programming languages such as Lisp, logical programming languages such as Prolog and script languages such as Perl is also provided. Within this course, students will write programmes in LISP, Prolog and PERL.

- **Distributed Systems**
  Architectures of distributed systems. Inter-process communications. Socket types and development platforms. Standard application solutions for distributed computing systems. Fundamentals of client - server programming. Node to node communication in Internet-based distributed computing systems. Middleware platforms (CORBA, JavaBeans, DCOM, .NET). Multi-agent systems in a distributed environment. Web Service technologies in distributed computer systems (architecture and implementation).

- **Laboratory Course**
  Supervised laboratory teamwork. Application of digitization. E-presentation. Copyright and authoring rights. Tools for digitization: scanning of images, resolution, image processing, bitmap and vector-oriented editing images.

- **Software Testing**
  Definitions and measures of reliability and confidentiality. Reliability and availability modelling. Detection and error codes for error correction. Design of reliable systems: transient versus permanent errors. Sources of errors in software, techniques for fault tolerance, reliability in VLSI devices, air control systems, telecommunication systems, and industrial control applications. Reliable systems for processing transactions. Internet access and software reliability. Models for software reliability. Methods for software reliability. Reliability in operational systems and data structures. Confidentiality in databases and distributed systems. Test design. Methods for generating tests. Automatic Test Pattern Generation (ATPG).

- **Team Project**
  This is an open subject where students can choose to work in a team project related to the latest advancements in the field of software engineering. Possible topics include the following areas: analysis and design of software systems, development of graphics applications, multimedia applications and e-business applications, testing, statistical data processing, metaphoric reasoning and reasoning by analogy.

- **Automata Theory and Formal Languages**
  This course includes the following topics: Automata theory, algorithm strategies, basic computer algorithms, geometric algorithms, cryptographic algorithms, parallel algorithms, distributed algorithms, basic algorithm analysis, advanced algorithm analysis, basic computability, complexity, complexity classes P and NP.

- **Operating System Security**
  This course includes the following topics: Encryption structure. Examples of encryption protocols. Encryption with secret keys. Public Key encryption. Cracking encrypted systems. Basic protective mechanisms in operating systems. Architecture protection systems in operating systems, authentication, access control: access lists, access control implementation (Unix, Java), Bell and La Padula model, mechanisms of operating systems for supporting MAC policies, security policies Clark-Wilson and Chinese wall. Weaknesses of protection in operating.

- **Master Thesis**
  This module enables students to transfer their skills and knowledge to research and make more complex task of the Master thesis. The module is designed to be fully practical and students to acquire the necessary knowledge and skills to approach writing the thesis. The module has unique return result-to enable students to write the master thesis with minimal difficulties, and with maximum efficiency. The course aims to improve research techniques and style of writing paper, taking account to stop illegal means, such as plagiarism and infringement of copyright, which are prohibited by the Statute of SEEU.

Elective courses

- **Human-Computer Interface**
  The purpose of this course is to provide students with knowledge of the principles of Human-Computer Interaction including, an evaluation of user interfaces task analysis, user-centered design, prototypes and conceptual models

and metaphors.

- **Advanced Web Programming**
  The purpose of this course is to provide students with the technological fundamentals, knowledge and skills related to a variety of web development frameworks that are used to implement and deploy web and mobile applications and services.

- **Interactive System Design**
  This course includes the following topics: Life-cycle stages: determining requirements, logical design, physical design and planning; interpersonal communication skills, interview and presentation; dynamics of group work; feasibility analysis and risk planning; project management, joint application development and structural examinations; rapid application development, prototype development; database design; evaluation of software packages, acquisition and integration; global and inter-organizational problems and integration of systems; professional codes and ethics.