UNIVERSITETI I EJL
ЈИЕ УНИВЕРЗИТЕТ
SEE UNIVERSITY

FACULTY

OF CONTEMPORARY SCIENCES AND TECHNOLOGIES

STUDY PROGRAMME FOR POSTGRADUATE STUDIES

(Master of Science)

NAME OF THE PROGRAMME:

SOFTWARE ENGINEERING STUDIES

INTRODUCTION

The Masters programme in Computer Sciences 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 learn to apply the acquired theoretical and technical skills both in the industry and the business sector.

The proposed fields are in line with the ICT-market needs as a whole, with a special emphasis on the local and regional market needs.

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 and three-year programmes.

The Masters programme in Software Engineering offers two subspecializations: subspecialization in Programming Languages and subspecialization in Operating Systems.

The first semester is common, and the students are required to choose one of the abovementioned subspecializations in the second semester and comply to the selected subspecialist line until the end.

Faculty of Contemporary Sciences and Technologies	
1. Institution/ Title issuing body	SEE University
2. Academic Institution	SEE University
3. Location of Instruction	SEEU Campus (Tetovo)
4. Programme accredited by	Accreditation Board
5. Final scholarly degree	M.Sc. in Computer Sciences
6. Programme	Software Engineering
7. Completion date/revision	September, 2007
8. Student group	2008/2009 Academic Year Generation

Programme objectives

- To provide students with opportunities to expand their analytical and research capacities.
- To equip students with theoretical and practical knowledge in Software Engineering.
- Students will perceive all the aspects of software development: needs and specification analysis, designing, coding, testing, integration and maintenance.
- To equip candidates for project work, both individual and group, which, by nature, may be scientific- research projects, developmental projects or internship.
- To provide background for further studies at doctoral and research level.

Learning outcomes	
Acquired knowledge	**Lectures/instruction, or teaching methods and assessment strategies used for acquiring and presenting of the outcomes**
A. Acquired knowledge and comprehension in: 1.Fundamental and advanced knowledge in Software Engineering. 2. Software development: needs and specifications analysis, designing, coding. 3.Fundamental and advanced knowledge in Programming Languages, their analysis and use in the development of different software solutions, or advanced knowledge in operating systems analysis and programming depending on the selected subspecialization. systems analysis and programming, 4. Managing big software projects and Processes. 5. General skills, such as analytical and critical thinking, team work and work in multicultural environments, planning and organizing etc.	**Lectures/Instruction** **Assessment** Within the course programme, assessment may be realized through the following breakdown: •Practical work realized successfully •Consultations •Realized projects •Seminar papers •Written exams etc.

Study programme structure, admission requirements, levels, modules, credits and awards

Instruction is performed in a full-time form. The programme courses are divided into obligatory and electives. The courses are divided into semestral units, each course is worth 6 credits, which is 180 classes of lecture duration. 60 credits are gained per year, which is about 1800 classes annually.

The draft programme in Masters Studies is organized within a duration of four semesters, so that it is also suitable for the candidates who are completing undergraduate three-year programmes.

Students are required to acquire all the credits according to the following plan-programme. Participation in lectures is obligatory and is a condition for passing the course. Each course has to be completed by satisfying the requirements set in the course syllabus.

At the end of 4 (fourth) semester students are also required to submit and defend a Masters Thesis.

CURRICULUM FOR POSTGRADUATE STUDIES IN SOFTWARE ENGINEERING

Semester 2	Credits	W/S	Lectures:	Tutorials:	Overall classes:	Group size
Research Methodology	6	W	15	30	180	50/20-30
Advanced Data Structures and Algorithms	6	W	15	30	180	50/20-30
Software Processes and Management	6	W	30	15	180	50/20-30
Seminar Paper	6	W	15	30	180	50/20-30
Elective	6	W	15	30	180	50/20-30
Total	**30**				**900**	

Subspecialization: Programming Languages

Semester 2	Credits	W/S	Lectures:	Tutorials:	Overall classes:	Group size
Software Analysis and Design	6	S	15	30	180	50/20-30
Programming Paradigms	6	S	15	30	180	50/20-30
Laboratory Course	6	S	15	30	180	50/20-30
Elective 1	6	S	15	30	180	50/20-30
Elective 2	6	S	15	30	180	50/20-30
Total	**30**				**900**	

Elective Courses:

1. Logical Programming
2. Functional Programming
3. Compilers and Interpreters
4. Programming Setting
5. Script Languages
6. Man-Computer Interaction etc.

Subspecialization: Operating Systems

Semester 2	Credits	W/S	Lectures:	Tutorials:	Overall classes:	Group size
Software Analysis and Design	6	W	15	30	180	50/20-30
Concurrent and Distributed Computing	6	W	15	30	180	50/20-30
Laboratory Course	6	W	15	30	180	50/20-30
Elective	6	W	15	30	180	50/20-30
Elective	6	W	15	30	180	50/20-30
Total	**30**				**900**	

Elective Courses:

1. Real-time Systems
2. System Programming
3. Fault- Tolerant Systems
4. Man-computer Interaction
5. Advanced Computer Architecture
6. Computer Organization Concepts etc.

Subspecialization: Programming Languages

Semester 3	Credits	W/S	Lectures:	Tutorials:	Overall classes:	Group size
Software Testing	6	W	15	30	180	50/20-30
Team Project	6	W	15	30	180	50/20-30
Elective 1	6	W	15	30	180	50/20-30
Elective 2	6	W	15	30	180	50/20-30
Elective 3	6	W	15	30	180	50/20-30
Total	**30**				**900**	

Elective Courses:

1. Network Programming
2. Object-oriented Programming
3. Automated and Formal Languages
4. Interactive System Design
5. Advanced Web-Programming
6. Visual Programming with Databases
7. Computer System Simulation and Design etc.

Subspecialization: Operating Systems

Semester 3	Credits	W/S	Lectures:	Tutorials:	Overall classes:	Group size
Software Testing	6	W	15	30	180	50/20-30
Team Project	6	W	15	30	180	50/20-30
Elective 1	6	W	15	30	180	50/20-30
Elective 2	6	W	15	30	180	50/20-30
Elective 3	6	W	15	30	180	50/20-30
Total	**30**				**900**	

Elective Courses:

1. Distributed Operating Systems
2. Operating System Security
3. System Programming in UNIX
4. Advanced Operating Systems
5. Networking

6. Networking and Multimedia Operating Systems etc.

Semester 4	Credits	W/S	Lectures:	Tutorials:	Overall classes:	Group size
Masters Thesis Writing	30				900	1
Total	**30**				**900**	

COURSE DESCRIPTION

SEMESTER 1

Research Methodology

This course covers different research methodologies in Computer Sciences and their combination: research assessment criteria in Computer Sciences; the nature of the research in Computer Sciences; practical pieces of advice for conducting research, as well as different research skills: reading, examining and assessing, as well as designing a research project; undertaking practical research tasks, preparing verbal and written presentations, research reports and writing etc.

Advanced Data Structures and Algorithms

The following topics are included in the course: algorithm analysis, recurrences and asymptotes, statistical analysis and accidental algorithms; overview of database structures (trees, heaps, hatch tables) and efficient sorting algorithms, searching and selection; database structures and online problems (balanced trees, branching database structures); algorithm design techniques: divide and rule, dynamic programming, greedy algorithms, amortization analysis; advanced database structures, including B-trees, binominal and Fibonacci heaps, representation of separate sums in database structures; graph problem algorithms such as breadth and depth-first search, MSTs and shortest path. Other possible topics: network flow, linear programming, string search, NP-completeness, approximation algorithms.

Software Processes and Programming

The course objective is to provide knowledge about: the techniques and programmes for software programme development, definition of processes, evaluation and process management, writing specifications and documentation and completing tests and evaluation. Specific methods and techniques for managing important activities and phases of the life cycle in the software development will be discussed and analyzed. This involves introduction to the techniques for software evaluation, instructions on how to systematically manage the needs and how to quantitatively inspect and detect the software defects.

SEMESTER 2

Subspecialization: Programming Languages

Software Analysis and Design

The course covers the different phases of the software system analysis and design. The topics covered are: software system analysis, software needs analysis, modeling tools and analysis and design methods, documentation development about the software specifications and needs, software testing and debugging, as well as software design development and testing documentation.

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 like Lisp, logical programming languages like Prolog and script languages like Perl is also provided. Within this course, students will write programmes in LISP, Prolog and PERL.

Subspecialization: Operating Systems

Software Analysis and Design

The course covers the different phases of the software system analysis and design. The topics covered are: software system analysis, software needs analysis, modeling tools and analysis and design methods, documentation development about the software specifications and needs, software testing and debugging, as well as software design development and testing documentation.

Concurrent and Distributed Computing

Concurrent and parallel systems taxonomy. Communication and synchronization, systems with many computers and many processors. Models of parallel and distributed computing. Design and analysis of algorithms for parallel and distributed systems. Basic techniques, classical problems. Complex parallel and distributed classes. Share-memory and message passing programming paradigms; parallel problem solving.

SEMESTER 3

Subspecialization: Programming Languages and Operating Systems

Software Testing

The course covers the software testing techniques: unit testing, integration testing, system testing, acceptability testing and regression testing. Knowledge about: testing environment, ability for planning and conducting testing, designing testing cases, the use of testing tools, ability for developing status reports on testing will be practised and required.

SEMESTER 4

Masters Thesis

Completion of the Masters Thesis which is in line with the University and Faculty Rulebook for Postgraduate Studies and Masters Thesis.