Master Program in Computer Science

Outline Descriptions of Modules
for
The MSc Study Plan

2006-2007

Department of Computer Science

I- The Compulsory Modules

(18 Credit Hours for Comprehensive exam path or 15 Credit Hours for thesis path)

Module Title	Module Number	Credit Hours
Algorithms and Complexity	750721	**3**
Advanced Operating Systems	750732	**3**
Artificial Intelligence	750751	**3**
Advanced Concepts in Database	750761	**3**
Fundamentals of Scientific Research	750791	**3**
Project	750798	**3**

750721, Algorithms and Complexity

3 hours per week, 3 credit hours, prerequisite: **none**

Teaching Method: 37 hours Lectures (2-3 hours per week) + 8 hours Seminars (1 per 2 weeks)

Aims: This module aims to introduce the concepts of analysis and design of algorithms and measure their complexity through different strategies.

Learning Outcomes:

On completion of this module, the student should:

- Understand different strategies of algorithms design.
- Be able to analyze different algorithms.
- Be able to measure the complexity of algorithms.
- Be able to choose the best algorithm for solving a problem.

Textbooks and Supporting Materials:

1. T.H. Cormen, C.E. Leiserson, R .L. Rivest, Introduction to Algorithms, PHI,
2. D. Kreher and D.Stinson: Combinatorial Algorithms: generation, enumeration and search, CRC Press, 1998.
3. Jean-Daniel Boissonnat, Mariette Yvinec, Algorithms Geometry,
4. M. R. Gerey, D. S. Jonhnson, Computers and Intractability: a guide to the theory of NP-completeness, W. H. Freeman, 1979
5. Frank Thomson Leighton, Introduction to Parallel algorithms and Architectures: Arrays, Trees, Hypercubes,
6. Mark de Berg, Marc van Kreveld, Mark Overmars, Computational Geometry,
7. A. V. Aho, J. E. Hopcroft, J.D. Ulman, The Design and Analysis of Computer Algorithms, Addison Wesley,
8. Michael Seul, Lawrence O'Gorman, Michael J. Sammon, Practical Algorithms for Image Analysis: Descriptions, Examples, and Code, Cambridge University Press,

Synopsis*:*

1. Revision of Design Methods: Divide and Conquer design, Dynamic Programming, Greedy algorithm etc. Master method, Order statistics, Dynamic Order Statistics, Red-Black trees, Binomial Heaps and Fibonacci Heaps, Data Structures for Disjoint Sets.
2. Matrix Chin Multiplication, Matrix Operations algorithms, Longest Common Subsequence Problems

3. Graph Algorithms, Minimum spanning Trees, Single-Source Shortest Paths, All-Pairs Shortest Paths, Maximum Flow Algorithms.
4. Sorting Networks, and complexity
5. Polynomials and DFT, FFT algorithms.
6. Combinatorial Algorithms, Number Theoretic Algorithms, String Matching algorithms,
7. Cryptography
8. Computational Geometry, Algorithmic Geometry and Complexity.
9. NP-Completeness, Non-deterministic algorithms
10. Exponential-time Techniques
11. Approximation algorithms
12. Parallel Algorithms.

Assessment: One 2-hours midterm exam (30%); Assignments (20%); Seminars (10%); 2-hours Final Exam (40%)

……..……

750732, Advanced Operating Systems

3 hours per week, 3 credit hours, prerequisite: **none**

Teaching Method: 37 hours Lectures (2-3 hours per week) + 8 hours Seminars (1 per 2 weeks)

Aims: The aim of this module is to provide advanced concepts in operating systems design. At advanced level, we will mainly study the architectural aspects and theoretical foundation of OS design, and an integrated approach to OS and Distributed Systems.
Important topics will include naming, security, remote procedure call, networks, concurrency, transactions, parallel computing, shared memory, message passing, scale and middleware.

Learning Outcomes:
On completion of this module, the student should:
- To accomplish system design with an integrated approach of modern software systems.
- To comprehend existing systems and to design new systems.
- To be accustomed with the modern trend of distributed systems.
- To design and develop modern operating systems to provide protection and security in the distributed systems and any concurrency systems.

Textbooks and Supporting Materials:
1. Bacon, Jean, *Concurrent Systems* Addison-Wesley, 2003
2. Bayer et. al .(eds.), Operating Systems An Advanced Course, Springer-Verlag, 1978.
3. Chow, R. Johnson, T., Distributed Operating Systems and Algorithms, Addison-Wesley, 1997.
2. Crowley, Charles, *Operating Systems: A Design-oriented Approach* Irwin 1997
3. Singhal, M., Shivaratri, N., Advanced Concepts in Operating Systems, McGraw-Hill, 1994.
4. Maekawa, et. al., Operating System Advanced Concepts, Benjamin-Cummings, 1987.
5. Nutt, Gary. *Operating Systems: A Modern Perspective, Lab Update 2e* Addison-Wesley 2002 ISBN 0-201-74196-2
6. Nutt, Gary. *Kernel Projects for Linux* Addison-Wesley 2000ISBN 0-201-61243-7
7. Stallings, William, *Operating Systems: Internals and Design Principles 3rd ed.* Prentice-Hall 1998
8. Tanenbaum, *Operating Systems: Design and Implementation 2nd ed.* Prentice-Hall 1997

9. Tanenbaum, *Modern Operating Systems 2nd ed*. Prentice-Hall 2001 ISBN 0-13-031358-0
10. Silberschatz & Galvin, *Operating System Concepts 5th ed.* John Wiley 1998

Usenet:
comp.os.*, comp.sources.unix, comp.unix.*, comp.windows.x

Technical Journals and other references :
* CACM, Computing Surveys, JACM, TOCS, SigOPS ,
* ACM's Computing Research Repository,
* ACM SIGOPS
* SOSP 2005
* HOTOS05
* OSDI 2004
* SOSP 2003

***Synopsis**:*
1. Overview of operating systems and Operating system principles
2. Concurrent Systems
3. Single Concurrent Actions
4. Concurrent Composite Actions
5. Distributed Systems
6. Security and Protection

Assessment: One 2-hours midterm exam (30%); Assignments (20%); Seminars (10%); 2-hours Final Exam (40%)

……..……

750751, Artificial Intelligence

3 hours per week, 3 credit hours, prerequisite: **none**

Teaching Method: 37 hours Lectures (2-3 hours per week) + 8 hours Laboratory (1 per 2 weeks)

Aims: The aim of this module is to introduce students to some of the basic ideas and concepts which underlie the development of artificially intelligent machine systems and to teach a programming language suited to the implementation of such systems, so that they are brought to a level from which they are able to pursue research in artificial intelligence.

Learning Outcomes:
On completion of this module, the student should:
- Have knowledge and understanding of knowledge-based system concepts.
- Have deeper knowledge and understanding of design philosophy behind rule-based systems.
- Be able to design and implement simple rule-based systems.
- Be able to consider the set of methods be needed to develop an intelligent system and have an appreciation of their advantages and limitations.
- Be able to use a set of tools to analyze intelligent tasks and be able to implement Prolog programs to solve them.

Textbooks and Supporting Materials:
1- P. H. Winston, Artificial Intelligence, 4th ed., Addison-Wesley,
2- L. Sterling, E. Shapiro, The Art of Prolog, MIT Press,
3- I. Bratko, Prolog Programming for Artificial Intelligence, 3rd ed. Addison-Wesley,

Synopsis*:*
1- Historical Overview: Definition of artificial intelligence (AI); Application areas; General problem solving versus specific knowledge; Complexity.
2- Heuristic Search: Uninformed versus informed search strategies; Formal properties of A*; Minimax game search; alpha-beta pruning.
3- Logic and Resolution: Knowledge representation; Propositional and predicate calculus; Inference rules; Clause form; Resolution strategies; Prolog and logic programming.
4- Uncertainty Reasoning: Probabilistic reasoning and Bayes theorem; Belief networks; Dempster-Shafer theory; Fuzzy logic.
5- Theory of Logic Programs: comparison with other paradigms; universal and existential quantification; facts; queries; logical variables; recursion; rules; Horn clauses; structured data.
6- Basic Prolog: execution model; declarative and procedural meaning; backtracking; arithmetic; list representation; negation as failure and difficulties; simple examples.
7- Prolog Programming and Techniques: input/output; meta-logical and extra-logical predicates; set predicates; cuts; program development and style; correctness and completeness; tracing and debugging.
8- Applications: Heuristic search and problem-solving strategies; game search; non-deterministic programming.

Assessment: One 2-hours midterm exam (30%); Assignments (20%); Seminars (10%); 2-hours Final Exam (40%)

……..……

750761, Advanced Concepts in Database

3 hours per week, 3 credit hours, prerequisite: **none**

Teaching Method: 37 hours Lectures (2-3 hours per week) + 8 hours Laboratory (1 per 2 weeks)

Aims: The goal of this module is to study some contemporary technologies in the database area that have been adopted in real applications and to survey products and applications that embody these technologies. Selected topics from the following will be covered: Object Oriented and Object Relational Databases; XML/Semi-structured Data Management; Temporal, Spatiotemporal, Time-Series, and Sequence Data Management; Distributed Databases; Information Retrieval Techniques; Advanced Query Optimization; Data Stream Systems; Main Memory Databases.

Learning Outcomes:
On completion of this module, the student should:
- Have knowledge and understanding of database technologies.
- Have deeper knowledge and understanding of more advanced topics is database.
- Be able to design and implement database systems.
- Be able to manage the database systems
- Be able to use different information retrieval techniques.

Textbooks and Supporting Materials:
1- Abraham Silberschatz , Henry F. Korth, and S. Sudarshan, Database System Concepts, 5th Edition, McGraw-Hill, May 2005

Research papers

Website(s): http://www.cs.yale.edu/homes/avi/db-book/db5/index.html

Synopsis:
1- Introduction: Database-System Applications, Purpose of Database Systems, View of Data, Database Languages, Relational Databases, Database Design, Object-Based and Semistructured, Data Storage and Querying, Transaction Management, Data Mining and Analysis, Database Architecture, Database Users and Administrators.
2- Relational Databases: Relational Model, SQL, Advanced SQL, Other Relational Languages, Database Design and the E-R Model, Relational Database Design, Application Design and Development.
3- Object-based Databases and XML: Object-Based Databases, XML.
4- Querying: Query Processing, Query Optimization.
5- Transaction Management: Transactions, Concurrency Control, Recovery System.
6- Data mining and Information Retrieval**:** Data Analysis and Mining, Information Retrieval.
7- System architecture: Database-System Architectures, Parallel Databases, Distributed Databases.
8- Others Topics: Advanced Application Development, Advanced Data Types and New Applications, Advanced Transaction Processing, Microsoft SQL Server.

Assessment: One 2-hours midterm exam (30%); Assignments (30%); 2-hours Final Exam (40%)
……..……

750791, Fundamentals of Scientific Research

3 hours per week, 3 credit hours, prerequisite: **none**

Teaching Method: 24 hours Lectures (1-2 hours per week) + 24 hours Seminars (1-2 hours per week)

Aims: This module aims to offer a grounding in various aspects of research and project management, from the most theoretical (philosophy of science), through the subject-specific (how to choose, refine and develop a research topic), to practical advice on undertaking research, including how to contribute to research, manage research projects, cope with the day-to-day research activity, etc. It covers material and advice on technical writing for the dissertation. Research seminars undertaken as part of the Research Project contribute to this module. The module also covers various aspects of Professional Skills and ethics as required in the IT industry and in Research and Development. The skills include team-work skills, industrial problem-solving, leadership skills, communication skills, presentation skills and preparation for job application and interview skills.

Learning Outcomes:
On completion of this module, the student should:
- Be prepared to undertake the Research Project, having been introduced to the skills and knowledge necessary to undertake the project.

- Have presented a research seminar to an audience of researchers.
- Have been prepared for some of the demands of, and skills required for; work in IT and IT-related industries.

Textbooks and Supporting Materials:

1- C. R. Kotheri, Research Methodology: Methods and Techniques, New Agw International, 2004
2- Anthony M. Graziano and M. L, Raulin, Research Methods: A process of Inquiry, Pearson International Edition, 2007
3- John W. Creswell, Research Design: Qualitative , Quantitative and Mixed Methods Approach, Sage Publications, 2003
4- Crawford, H. J. and Christensen, L. J., DFeveloping Research Skills, 4th edition, Allyn and Bacon, 2001
5- Donald H. McBurney, Research Methods, Wadsworth, 2000
6- Lawrence R. Frey, Carl H. Botan, and Gary L. Kreps, Investigating Communication: An Introduction to Research Methods, Allyn and Bacon, 2000
7- Victor O. K. Li, Hints on Writing Technical Papers and Making Presentations, IEEE Transaction on Education, Vol. 42, N0. 2, May 1999.
8- Compton R. T. Jr., Fourteen Steps to a Clearly Written Technical Paper, IEEE Circuits and Device Magazine, Sep. 1992

Web Sites

1- www.cs.iastate.edu/~honavar/grad-advice.html
2- http://research.microsoft.com/~simonpj/papers/giving-a-talk/giving-a-talk.html
3- http://owl.english.purdue.edu/
4- http://owl.english.purdue.edu/handouts/index.html
5- http://www.idt.mdh.se/kurser/ct3340/ht07/

Synopsis:

1- Research Skills and MSc project management: Introduction to research in science; Research methods and Creative thinking; Management of the MSc project, including managing the academic year, relationship with supervisor and interaction with research groups; Requirements of an MSc research project; Research presentations; Requirements of a good dissertation; Technical writing skills; Compare and contrast the scientific approach with other ways of obtaining knowledge and understand how the methods differ with regard to causality and generalizability; Compare the major research designs and discuss the strengths and weaknesses of each; Articulate the advantages of the scientific approach to practice; Define basic statistical terms and concepts, and discuss the concepts of measurement, sampling and data collection;
2- Explain how the scientific approach may be affected by ethics, and issues relating to diversity, minority status or oppression; Describe how the scientific approach can be used to test the efficacy of scientific research/projects; Design and conduct a study intended to improve practice.
3- Employ appropriate professional journal styles and formats when writing. Objectively critique published studies in scientific literature.

Assessment: There is no mid term exam for this module, but active participation is required, and students will need some of the material to succeed in the Research Project. The research seminar and assignment work are required to be assessed to provide feedback on performance, whose weight is 60%. The final written 2-hours exam weights (40%).

……..……

750798, Project

3 hours per week, 3 credit hours, prerequisite: **none**

Teaching Method: 48 hours supervised research work

Aims: This module aims to provide students with the ability to do scientific research work under the supervision of one of the lecturers.

Learning Outcomes:
On completion of this module, the student should:

- Understand and use the concepts of conducting scientific research.
- Be able to do self-learning and knowledge gathering
- Be able to write a project document efficiently
- Be able to present the scientific research to audience

Assessment: The project defense and the quality of the report document worth (100%)

……..……

II- The Elective Modules

(15 Credit Hours for Comprehensive exam path or 9 Credit Hours for thesis path)

Module Title	Module Number	Credit Hours
Parallel Programming Languages	750711	**3**
Fuzzy Logic in Computer Science	750722	**3**
Genetic Algorithms and Neural Networks	750723	**3**
Parallel Computer Architecture	750731	**3**
Mobile and Distributed Computing	750741	**3**
Theory of Concurrency	750742	**3**
Natural Language Processing	750752	**3**
Machine Learning	750753	**3**
Data Mining and Data Warehousing	750762	**3**
Multimedia Systems Technology	750771	**3**
Multimedia and the Web	750772	**3**
Formal Methods in Software Engineering	750781	**3**
Software Process	750782	**3**
Software Maintenance	750783	**3**

750711, Parallel Programming Languages

3 hours per week, 3 credit hours, prerequisite: **none**

Teaching Method: 30 hours Lectures (2 hours per week), 7 hours Seminars (1 per 2 weeks), 8 hours Laboratories (1 per 2 weeks)

Aims: This module aims to cover a variety of paradigms and languages for programming parallel computers. Topics covered in depth include parallel programming techniques with their implementations in a parallel environment; shared-memory and message-passing models, process synchronization and data sharing, communication; converting sequential algorithms into equivalent parallel algorithms; improving performance of parallel algorithms. Several tools for debugging and measuring the performance of parallel programs will be introduced and some example languages will be covered.

Learning Outcomes:
On completion of this module, the student should be able to:

- Know a variety of paradigms and languages for programming parallel computers.
- Know how to convert sequential programs for single processor computers into faster working programs that give the same results when run on parallel computers, especially those with globally shared address spaces.
- Understand the fundamental aspects of parallel processing.
- Implement the new concepts of programming languages in parallel programs.
- Be familiar with performance measures of parallel programs and the fundamental limits on their speedup.
- Comprehend the distinction between different programming models.
- Understand the theoretical limitations of parallel computing such as intractability.
- Design and analyze simple parallel algorithms.
- Write efficient parallel application programs.
- Write programs using thread programming methods for a shared memory computer model.

Textbooks and Supporting Materials:

1- Barry Wilkinson, Michael Allen. Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers, 2ed, Pearson/Prentice Hall, 2005. ISBN 0-13-140563.
2- Salim G. AKI, Parallel Computation Models and Methods, Prentice Hall, 1997.
3- Michael J. Quinn. Parallel Programming in C with MPI and OpenMP, McGraw Hill (2004), ISBN 0-07-282256-2
4- Shirley A. Williams, Programming Models for Parallel Systems, John Wiley, 1990
5- P. I. Fleming (editor), Parallel Processing in Control: the transputer and other architectures, Peter Peregrines ltd, 1988.
6- Joel M. Crichlow, An Introduction to Distributed and Parallel Computing, 2nd edition, Prentice Hall, 1997.
7- Michael J. Quinn, Designing Efficient Algorithms for Parallel Computers, McGraw Hill, 1987.
8- Foster, I. T., Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering, (ISBN 0-201-575-949), Addison-Wesley 1995. (Very good on message-passage-passing style of programming)
9- M. L. Scott, Programming Languages Pragmatics, Morgan Kaufman Publishers, 2000
10- M. E. C. Hull, D. Crookes, D. J. Sweeney (editors), Parallel Processing: the Transputer and its Applications, Addison Wesley, 1994

11- Culler, D. E. and Singh, J.P., with Gupta A., Parallel Computer Architecture: a hardware/software approach, (ISBN 1-55860-343-3), Morgan Kaufmann 1999. (Very good book on architecture level)
12- Hockney, R. W. and Jesshope, C.R., Parallel Computers 2, (ISBN 0-862-748-124), Adam Hilger 1988. (A good, albeit old-fashioned reference text for fundamental techniques)
13- David E. Culler and Jaswinder Pal Singh, with Anoop Gupta. Parallel Computer Architecture: A Hardware/Software Approach, Morgan Kaufmann, 1998. ISBN: 1-55860-343-3.
14- George Almasi and Allan Gottlieb, Highly-Parallel Computing, 2nd Edition, Benjamin-Cummings, 1994.
15- Ted G. Lewis, Hesham El-Remini, Introduction to Parallel Computing, Prentice Hall 1992.
16- Grama, A. Gupta, G. Karypis and V. Kumar. Introduction to Parallel Computing (2nd edition), Addison Wesley (2002), ISBN 0-201-64865-2.
17- H. El-Rewini and T. G. Lewis. Distributed and Parallel Computing, Manning (1997), ISBN 0-13-795592-8.
18- R. J. Barlow and A. R. Barnett, Computing for Scientists: Principles of Programming with FORTRAN 90 and C++, John Wiley and Sons, 1998.
19- Andrei Alexandrescu, Modern C++ Design: Generic Programming and Design Patterns Applied, Addison-Wesley, 2001
20- Robert Robson, Using the STL: The C++ Standard Template Library, Springer, 1997.
21- Bregory V. Wilson and Paul Lu (editors), Parallel Programming using C++, MIT Press, 1996.
22- Thuan Q. Pham and Pankaj K. Garg, Multithreaded Programming with Windows NT, Prentice Hall, 1995.
23- Gregory V. Wilson, Practical Parallel Programming analysed, MIT Press, 1995.
24- Foundations of Multithreaded, Parallel, and Distributed Programming, Addison-Wesley, 2000
25- E. V. Krishnamurthy, Parallel Processing Principles and Practice, Addison Wesley, 1989

*** Plus some Research Papers on the topics**

***Synopsis**:*

1- An Introduction to parallel programming languages: specifying parallel processes;
2- Categories of parallel programming models: sequential processing, Array processing, pipeline processing, shared memory processing, message passing, Data Parallel programming, functional programming, logic programming, object-oriented programming;
3- Software architectures: Semaphores, Monitors, Remote Method Invocation, the Ada rendezvous, message passing semantics;
4- Multithreaded programming: Threads, Synchronization techniques, Java threading model, Applications
5- Multiple process programming: Sockets, Messages, Applications, Client/Server models
6- Multiple process programming: CORBA
7- RMI: Basic principles, Techniques for using
8- MPI: Basic message, Synchronization, first MPI program, Scatter/gather, Broadcast messages, Groups/contexts, I/O
9- Parallel programming languages and algorithms: parallel language and algorithm design for the array processor: Actus; Von Neumann-type languages: Concurrent Pascal, Communicating Sequential Processes (CSP) and Occam, Distributed Processes (DP), Ada, Linda, C++, Java; Non-Von Neumann-type languages: functional programming (FP), Lisp.
10- The transputer implementation of Occam; Control application of Transputers
11- Detection of parallelism within expressions, parallelism in Tree structures, determining parallelism between blocks of programs,

12- Restructuring for Parallel Performance: Parallelising Compilers; Loop Transformations; Data Transformations; Dependence Compiler Strategies; Analysis; Reducing Parallelism; Parallelizing serial programs
13- Examples of Parallel Algorithms: Sorting and searchning algorithms; Cyclic Reduction; Iterative Algorithms (Jacobi, Gauss-Seidel and Red-Black Orderings); Divide-and-Conquer Algorithms

Assessment: One 2-hours mid-term exam (30%); Assignments (10%) (two assignments on parallel models of computation); writing a research paper / presentation / final project (20%); 2-hours Final Exam (40%).

……..……..……

750722, Fuzzy Logic in Computer Science

3 hours per week, 3 credit hours, prerequisite: **none**

Teaching Method: 37 hours Lectures (2-3 hours per week) + 8 hours Seminars (1 per 2 weeks)

Aims: This module aims to introduce the concepts of fuzzy logic and its applications in database systems, data structures, AI, and pattern recognition.

Learning Outcomes:
On completion of this module, the student should:
- Understand the concepts of fuzzy logic.
- Be able to use fuzzy logic in different areas of applications.
- Be able to give reasoning about the problem solutions.

Textbooks and Supporting Materials:
1. Bart Kosko, Fuzzy Engineering, Prentice Hall International, 1998.
2. Timothy Ross, Fuzzy Logic with Engineering Applications, McGraw Hill, 1995.
3. Klir and Yuan, Fuzzy Sets and Fuzzy Logic: Theory and Applications, Prentice Hall, 2000.
4. Frederick E. Petry and Patrick Boso, Fuzzy Databases: Principles and Applications (International Series in Intelligent Technologies, 5), Kluwer Academic Publishers, 1995.
5. A. Yazici, Fuzzy Databases Modeling, Springer Publisher, 2004.
6. Atanassov, K Physica-Verlag, Intuitionistic Fuzzy sets: Theory and Applications, New York, 2000.
7. R.R.Yager and L.A.Zadeh, An Introduction to Fuzzy Logic Applications in Intelligent System, Kluwer Academic Publisher, 1992.
8. Didier Dubois, Henri Prade and Ronald R. Yager, Fuzzy Information Engineering A Guided Tour of Applications, John Wiley and Sons, 1997.
9. Pal and Dutta Majumder, Fuzzy Pattern Recognition, Willy Eastern Publications, 1999.
10. Z.Pawalak, Rough Sets: Theoretical Aspects of Reasoning about Data, Kluwer Academic Publisher, 2002.
11. Inuiguchi, Masahiro, Tsumoto, Shusaku, and Hirano, Shoji, Rough Set Theory and Granular Computing Series: Studies in Fuzziness and Soft Computing, Springer 2003.

Synopsis*:*
1- Imprecise and Incomplete data/information, Uncertainty, Vagueness, Fuzziness, fuzzy hedges, Approximate Reasoning Concept.

2- Fuzzy Sets, Rough Sets and Systems, Examples, Various Fuzzy Operations, Graphical Representations.
3- Fuzzy Number and various operations on them, Fuzzification and De-fuzzification.
4- Fuzzy Relations and various applications of fuzzy relations, fuzzy graphs.
5- Fuzzy Rule-Based System Centre of Gravity method.
6- Applications of Fuzzy Logic in Database System.
7- Applications in Data Structures.
8- Applications in Pattern Recognition.
9- Applications in AI

Assessment: One 2-hours midterm exam (30%); Assignments (15%); Seminars (15%); 2-hours Final Exam (40%)

……..……

750723, Genetic Algorithms and Neural Networks

3 hours per week, 3 credit hours, prerequisite: **none**

Teaching Method: 37 hours Lectures (2-3 hours per week) + 8 hours Seminars (1 per 2 weeks)

Aims: By studying this module, students will gain an understanding of two of the principal components of 'soft' artificial intelligence, namely Genetic Algorithms (GA) and Artificial Neural Networks (ANN). Artificial neural networks attempt to capture the essential processing mechanism, which underlies the human brain, by manipulating a network of interconnected nodes, each with fairly simple processing capabilities. ANN has been used successfully for many classification applications, such as pattern recognition, and cluster analysis. GA uses the principle of natural selection to 'evolve' artificially a population of candidate solutions through simulated reproduction and mutation. GA has been used successfully for engineering optimization tasks and for computational problems. This module will provide students with an appreciation of theoretical issues and practical applications of variety of ANN architectures, including basic back-propagation, self-organizing maps and radial basis function networks, and knowledge of the GA, including data representation, genetic operators and properties of GA. Both GA and ANN will be studied in practical laboratory sessions. For the designing, training and simulating of the ANN. MATLAB NN GA Toolbox will be used.

***Learning Outcomes*:**
On completion of this module, the student should be able to:
- Evaluate and implement a variety of ANN architectures.
- Evaluate and implement the basic GA.
- Analyze and use 'soft-computing' problem-solving techniques.
- Design, train, critically evaluate and implement ANN for solving 'real-world' problems (use of MATLAB and UCI repository data bases).

Textbooks and Supporting Materials:
1- P. Picton, Neural Networks, Palgrave, 2000.
2- I. Nabney, NETLAB Algorithms for Pattern Recognition, Springer, 2002.
3- A. Zilouchian and M. Jamshidi, Intelligent Control Systems Using Soft Computing Methodologies, CRC Press, 2001.

4- Z. Michalewicz, Genetic Algorithms and Data Structures = Evolution Programs, Springer-Verlag, 1999.
5- M. Mitchell, An Introduction to GA, MIT Press, 1996.
6- A. Engelbrecht, Computational Intelligence, John Wiley and Sons, 2002.
7- S. Hykin, Neural Networks, Prentice-Hall 1999.

*** Plus some research papers on the topics**

***Synopsis**:*
1- Introduction to AI - Search Methods.
2- ANN - Single-Layer Perceptions; ADALINE; Perception Learning.
3- Multi-Layer Feed Forward NN - Supervised Learning; Back propagation.
4- Unsupervised and Competitive Learning - Kohonen's Self Organising Maps (SOM); Radial Basis Function Networks.
5- Introduction to Genetic Algorithms - GA Terminology and Operators (crossover, mutation, inversion)
6- Theory of GA - Schema properties; Implicit Parallelism; GA - Evolutionary Computing.
7- Selection, Replacement and Reproduction Strategies ('roulette wheel', elitism) - Fitness proportional selection; Premature convergence; Representation.
8- GA - advantages, disadvantages and applications; GA and NN.

Assessment: One 2-hours midterm exam (30%); Seminars (10%); Assignments (20%) (this will include designing, training, critical evaluation and implementation of ANN for solving real-world problem (with the use of MATLAB NN GA Toolbox eg. UCI repository data bases)); 2-hours Final Exam (40%).

……..……

750731, Parallel Computer Architecture

3 hours per week, 3 credit hours, prerequisite: **none**

Teaching Method: 30 hours Lectures (2 hours per week), 7 hours Seminars (1 per 2 weeks), 8 hours Laboratories (1 per 2 weeks)

Aims: The aims of this module are to introduce the main classes of parallel computer architectures and their distinguishing characteristics; examine some examples of particular architectures that were important in the development of parallel computers; examine some examples of recent parallel architectures; introduce some main principles of parallel programming and the effects of the architecture on algorithm design; understand how to write efficient parallel programs.

Learning Outcomes:
On completion of this module, the student should be able to:
- Understand the main classes of parallel architectures and their distinguishing characteristics and relative merits.
- Understand the major characteristics of some examples of the main architectural classes.
- Be aware of various styles of parallel hardware architecture and the structure and operation of their major components.

- Understand the impact of parallel computer architecture on software design and performance.
- Understand the fundamental aspects of parallel processing
- Appreciate the nature of a small number of performance-critical applications.
- Comprehend the distinction between message-passing (process-based) and data-sharing (thread-based) programming models
- Be capable of performance analysis of (competing designs for) hardware components for parallel systems.
- Understand the link between source-code and hardware behaviour, and be capable of analysing the execution performance of small fragments of data-parallel code in terms of various classes of overheads associated with parallel execution.
- Be familiar with performance measures for parallel systems.
- Understand the theoretical limitations of parallel computing such as intractability.
- Design and analyze simple parallel algorithms.
- Write efficient parallel application programs

Textbooks and Supporting Materials:

1- Foster, I. T., Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering, (ISBN 0-201-575-949), Addison-Wesley, 1995. (Very good on message-passage-passing style of programming, but has useful chapters on other aspects of the course)

2- Culler, D. E. and Singh, J.P., with Gupta A., Parallel Computer Architecture: a hardware/software approach, (ISBN 1-55860-343-3), Morgan Kaufmann 1999. (Very good book on architecture level)

3- Hockney, R. W. and Jesshope, C.R., Parallel Computers 2, (ISBN 0-862-748-124), Adam Hilger, 1988. (A good, albeit old-fashioned reference text for fundamental techniques)

4- Barry Wilkinson, Michael Allen. Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers, 2nd ed., Pearson/Prentice Hall, 2005. ISBN 0-13-140563.

5- Jessen Havill, An Introduction to Linux at Denison, 2003.

6- David E. Culler and Jaswinder Pal Singh, with Anoop Gupta. Parallel Computer Architecture: A Hardware/Software Approach, Morgan Kaufmann, 1998. ISBN: 1-55860-343-3.

7- George Almasi and Allan Gottlieb, Highly-Parallel Computing, 2nd Edition, Benjamin-Cummings, 1994.

8- Michael J. Quinn. Parallel Programming in C with MPI and OpenMP, McGraw Hill, 2004, ISBN 0-07-282256-2

9- Grama, A. Gupta, G. Karypis and V. Kumar. Introduction to Parallel Computing, 2n edition, Addison Wesley, 2002, ISBN 0-201-64865-2.

10- H. El-Rewini, T.G. Lewis, Distributed and Parallel Computing, Manning, 1997, ISBN 0-13-795592-8.

11- B. Wilkinson, Computer Architecture Design and Performance, Prentice Hall, 1991

12- Dezso Sima, Terence Fountain, Peter Kacsuk, Advanced Computer Architectures A design Space Approach, Addison Wesley, 1997

*** Plus some Research Papers on the topics**

Synopsis*:*

1- Computer Organization for parallel and distributed computing: Pipeline and vector processors, Multicomputers and computer networks, multiprocessors: Synchronization, Interprocess communication;

2- Massively parallel architecture: Associative processors, Array Processors (SIMD), Large-scale MIMD; Non-von Neumann-type computers: data-flow machines, reduction machines
3- Introduction on Parallel Architectures: Taxonomy of systems, SIMD, MIMD, systolic arrays. Parallel programming paradigms. Applications using a multiple processor parallel architecture.
4- SIMD Array Processors: Example architectures: Illiac IV, AMT DAP, Connection Machine, systolic arrays.
5- Interconnection Networks: Basic characteristics, routing functions, network topologies, dynamic (switched) networks (cross-bar, bus, Benes networks, shuffle-exchange networks). Example: Connection Machine.
6- MIMD Shared-Memory Architectures: shared memory multi-processors: Example architectures: BBN Butterfly, DEC AlphaServer 8000 as a symmetric multiprocessor architecture.
7- MIMD Message-Passing Architectures: The Transputer as a building block, complete machines using Transputers, hypercube machines (Cosmic Cube, Intel Hypercube and hypercubes using Transputers).
8- Programming Parallel Machines: Parallel Computation Models, writing parallel programs for abstract machines, the divide-and-conquer paradigm and some example algorithms, the doubling paradigm and example algorithms, programming real machines (SIMD, shared-memory MIMD, and message-passing MIMD)
9- Performance Measures: Granularity, Speed Up, Efficiency, Cost

Assessment: One 2-hours midterm exam (30%); Assignments (10%) (two assignments on parallel architectures and possible computations on them); Writing a research paper/Presentation/Final project (20%); 2-hours Final Exam (40%)

……..……

750741, Mobile and Distributed Computing

3 hours per week, 3 credit hours, prerequisite: **none**

Teaching Method: 37 hours Lectures (2-3 hours per week), 8 hours Seminars (1 per 2 weeks)

Aims: The goal of this module is to provide an in depth understanding of the fundamental problems in the area of mobile computing and study the existing and proposed solutions for these problems from both research and development perspective. Several topics including wireless communication, location management and mobility tracking, software engineering for mobile computing and applications are covered in this module.

***Learning Outcomes*:**
On completion of this module, the student should:
- Understand the area of mobile computing.
- Have knowledge on some aspects of software engineering for mobile computing.
- Be able to do a research work in mobile computing.

Textbooks and Supporting Materials:
I. Textbooks
1. K. S. Gupta, F. Adelstein, G. Richard and L. Schweibert, Fundamentals of Mobile and Pervasive Computing, McGraw-Hill, 2004.

2. J. Schiller, Mobile Communications, Second edition, Addison Wesley, 2003.
3. Bhaskar Krishnamachari, Networking Wireless Sensors, Cambridge Press, 2005.
4. C. E Perkins, Ad-Hoc Networking, Addison Wesley, 2001.
5. Formal Methods for Mobile Computing, 5th International School on Formal Methods for the Design of Computer, Communication, and Software 2005, Advanced Lectures Series: Lecture Notes in Computer Science, Vol. 3465.
6. Pan, Yi and Xiao, Yang, Design and Analysis of Wireless Networks, Wireless Networks and Mobile Computing, Volume 1, (Georgia State University) Pan, Yi - Series Editor, 2005.

II. Selected research Papers

III. Selected Mobile Programming Environments.

Synopsis*:*

1. Introduction to Mobile and Pervasive Computing: Mobile and wireless networks (cellular, ad hoc, sensor based); Applications (data broadcasting, context-aware); Challenges.
2. Mobile Networking: Mobile-IP; Ad-Hoc Networks; Sensor Networks; Wireless TCP; Session Mobility.
3. Mobility Management: Location management scheme; Handoff schemes.
4. Software Engineering for Mobile Computing: Mobile Computing Models; Formal methods for Mobile Computing: Software Architectures for mobile Systems, Languages.
5. Mobile Applications and Services: Mobile Agents; Transcoding and Proxy Architecture; Wireless Web and WAP; Peer-to-Peer Computing in Mobile Ad-Hoc Environment

Assessment: One 2-hours midterm exam (30%); Assignments (20%); Seminars (10%); 2-hours Final Exam (40%)

……..……

750742, Theory of Concurrency

3 hours per week, 3 credit hours, prerequisite: **none**

Teaching Method: 37 hours Lectures (2-3 hours per week), 8 hours Seminars (1 per 2 weeks)

Aims: This module aims to teach mathematical foundations to understand *models* of concurrent programs and reactive systems. The module covers basic techniques to describe the form and meaning of program terms and to reason about them (proving concurrent programs correctness). The module will be accompanied by pencil-and-paper as well as computer-aided verification exercises.

***Learning Outcomes*:**

On completion of this module, the student should:

- Understand the mathematical foundations of concurrent programs.
- Have knowledge on proving concurrent programs correctness.
- Be able to write concurrent programs and prove their correctness.

Textbooks and Supporting Materials:

I. Textbooks

1- Michael Huth and Mark Ryan, Logic in Computer Science Modeling and Reasoning about Systems, Cambridge Press, 2004.

2- Fokkink, Wan, Introduction to Process Algebra, Series: Texts in Theoretical Computer Science. An EATCS Series, 2000, VIII, 163 p. 11 illus.
3- Robert Milner, Communicating and Mobile Systems: The Π-Calculus,
4- Formal Methods for Mobile Computing, 5th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-Moby 2005, Bertinoro, Italy, April 26-30, 2005, Advanced Lectures, Series: Lecture Notes in Computer Science, Vol. 3465
5- Rewriting Techniques and Applications, 11th International Conference, RTA 2000, Norwich, UK, July 10-12, 2000 Proceedings, Series: Lecture Notes in Computer Science, Vol. 1833, Bachmair, Leo (Ed.)
2000, X, 275 p.
6- W. Reisig, G. Rozenberg (Eds.), Lectures on Petri Nets I: Basic Models, Advances in Petri Nets, Lecture Notes in Computer Science, Vol. 1491, Springer-Verlag, 1998.
7- Jörg Desel, Wolfgang Reisig, Grzegorz Rozenberg (Eds.), Lectures on Concurrency and Petri Nets, Advances in Petri Nets, Lecture Notes in Computer Science, Vol. 3098, Springer-Verlag, 2004, ISBN: 3-540-22261-8.

II. Selected Research papers
III. Selected Software Tools.

Synopsis*:*
1. Concurrency: Concurrent Programs; Non determinism, Synchronization, Communication; Correctness of concurrent programs.
2. Properties of concurrent programs: Liveness; Safety.
3. Models of concurrency: Interleaving; True concurrency.
4. Labeled Transition Systems: Transition Systems and notion of equivalence; Sequential processes and Bisimulation; Concurrent processes and Bisimulation; Basics on mobile processes (Π calculus); Reasoning about mobile processes.
5. Petri Nets: Generalities; Basic formal definitions; Petri nets properties; Verification techniques; High level Petri nets.
6. Rewriting Logic: Generalities; Basic formal definitions; Rewriting logic as a Semantics for true concurrency; Proof techniques.

Assessment: One 2-hours midterm exam (30%); Assignments (20%); Seminars (10%); 2-hours Final Exam (40%)

……..……

750752, Natural Language Processing

3 hours per week, 3 credit hours, prerequisite: **none**

Teaching Method: 37 hours Lectures (2-3 hours per week), 8 hours Seminars (1 per 2 weeks)

Aims: This module provides an introduction to the field of Natural Language Processing (NLP) - the creation of computer programs that can understand, generate, and learn natural language. Natural language understanding will be used as a vehicle to introduce the three major subfields of NLP: syntax (which concerns itself with determining the structure of a sentence), semantics (which concerns itself

with determining the explicit meaning of a single sentence), and pragmatics (which concerns itself with deriving the implicit meaning of a sentence when it is used in a specific discourse context).
The module also introduces the theory and techniques that are used in building computer systems that can "understand" a natural language such as English. The concentration is on the four main topics of syntax, parsing, semantics and semantic processing. The formalisms are used to capture syntactic and semantic knowledge and the algorithms that can use that knowledge.

Learning Outcomes:
On completion of this module, the student should be able to:

- Become aware of the issues involved in various forms of knowledge representation (for syntactic, semantic, pragmatic and world knowledge) that are needed for language processing.
- Become aware of the issues involved in various forms of machine reasoning (parsing, semantic translation, disambiguation, inference drawing) that are needed for language processing.
- Learn how to use the theoretical knowledge they gain in the construction of simple applications systems.

Textbooks and Supporting Materials:
The material will be based on published papers and textbooks. Students are expected to read and understand these.

1- J. Allen, Natural Language Understanding, 2nd ed., Benjamin/Cummings, 1994
2- G. Gazdar and C. Mellish, Natural Language Processing in Prolog, Addison Wesley, 1989
3- F. C. N. Pereira and S.M. Shieber, Prolog for Natural Language Analysis, Stanford University: Center for the Study of Language and Information, 1987
4- Gazdar, Klein, Pullum and Sag, Generalized Phrase Structure Grammar, Blackwell, 1985

Synopsis*:*

1- Introduction to Natural Language Processing (NLP)
2- Syntax: grammar in computational linguistics.
3- Parsing: simple parsing: efficient parsing algorithms ; left-corner and chart parsing; implementation in Prolog.
4- Semantics and semantic processing: compositional semantics; representation - logic and lambda calculus; semantic processing in Prolog; quantification.

Assessment: One 2-hours midterm exam (30%); Assignments (20%); Seminars (10%); 2-hours Final Exam (40%).

……..……

750753, Machine Learning

3 hours per week, 3 credit hours, prerequisite: **none**

Teaching Method: 37 hours Lectures (2-3 hours per week), 8 hours Seminars (1 per 2 weeks)

Aims: The development of "intelligent" software systems is an area that is becoming increasingly important in applied Computer Science. One view of such systems postulates that truly intelligent machines must include the ability to learn from experience and observation. The purpose of this module is to introduce students to some prominent "Machine learning" techniques. These allow the development of software systems that are capable of adaptive behaviors in complex real-world environments. Some examples are programs that assist users or adapt to their behaviors; identify interesting patterns in databases- like those generated by the Human Genome Project; and predict stock-market movements. In this module students will learn symbolic learning (decision and regression trees), learning using neural-networks, and probabilistic modeling techniques.

Learning Outcomes:
On completion of this module, the student should be able to:

- Understand some prominent "Machine learning" techniques.
- Understand symbolic learning and learning using neural-networks.
- Develop "intelligent" software systems that are capable of adaptive behaviors in complex real-world environments.
- Develop systems that learn from experience and observation.
- Be involved in the research, development and exploitation of machine learning and adaptive computing.

Textbooks and Supporting Materials:

1. Thomas M. Mitchell, Machine Learning, McGraw Hill, 1997.
2. I. Witten and E. Frank, Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufman, 1999.
3. Richard S. Satton, Aderw G. Barto, Reinforcement Learning, 1998.
4. Judea Pearl, Probabilistic Reasoning in Intelligent Systems, 1995

Synopsis*:*

1- Machine learning is concerned with the design of algorithms that, rather than encoding explicit instructions or programs for the solution of specific problems, encode inductive mechanisms whereby solutions to broad classes of problems may be derived from examples. Whether the approach is inspired by biology (e.g., artificial neural networks) or by cognitive psychology (e.g., traditional AI), machine learning aims at building computer systems capable of adapting to new tasks and learning from their experience.

2- Applications of machine learning and adaptive computing techniques in business and commerce. This is matched by research, which is directed at the provision of a new class of intelligent computer systems. The development of this area is dependent on the availability of a pool of personnel who are able both to understand and appreciate research findings and to use these to implement appropriate solutions to practical problems.

3- Topics in Evolutionary Computing, Neural Networks, Knowledge Systems, and Inductive Methods, and a series of smaller topics in Reinforcement Learning, Artificial Life, Agent-Based Systems, and Organizational and Business Aspects of IT. All topics cover the techniques associated with each area together with their applications.

4- A research project in this subject may provide a valuable opportunity for students to gain work experience and to consider their future direction.

Assessment: One 2-hours midterm exam (30%); Assignments (20%); Seminars (10%); 2-hours Final Exam (40%)

……..……

750762, Data Mining and Data Warehousing

3 hours per week, 3 credit hours, prerequisite: **none**

Teaching Method: 37 hours Lectures (2-3 hours per week), 8 hours Seminars (1 per 2 weeks)

Aims: The data warehousing part of the module aims to give students a good overview of the ideas and the techniques, which are behind recent developments in the data warehousing and OnLine Analytical Processing (OLAP) fields, in terms of data models, query languages, conceptual design methodologies, and storage techniques. Laboratory sessions will ground the abstract notions on practical cases and tools.
The data mining part of the module aims to motivate, define and characterize data mining as a process; to motivate, define and characterize data mining applications; to survey, and present in some detail, a small range of representative data mining techniques and tools. Laboratory sessions will ground the abstract notions on practical cases and tools.

Learning Outcomes:
On completion of this module, the student should be able to:
- Understand the techniques behind the recent development in data warehousing and data mining.
- Understand query languages and conceptual design methodologies.
- Practice on different tools of data warehousing and data mining.
- Design small projects with data mining and data warehousing.

Textbooks and Supporting Materials:
1. M. Jarke, M. Lenzerini, Y. Vassiliou, P. Vassiliadis (ed.), Fundamentals of Data Warehouses, Springer-Verlag, 1999.
2. Ralph Kimball, The Data Warehouse Toolkit, Wiley 1996.
3. I. Witten and E. Frank, Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufman, 1999. (This is the one that lectures notes are most closely based on.)
4. J. Han and M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufman, 2000. (This is more database-centred, in contrast to Witten and Frank, who takes a machine-learning viewpoint of data mining. It is also useful in covering data warehouses too, to some extent.)
5. D. Hand, H. Mannila and P. Smyth. Principles of Data Mining, MIT Press, 2001. (This takes yet another viewpoint on data mining, viz., the statistical one. In this sense, it is the least related to the approach followed in this part of the course.)
6. M. H. Dunham. Data Mining: Introductory and Advanced Topic. Prentice Hall, 2003. (This has yet another slight shift in emphasis, as it more or less favours an algorithmic viewpoint and is, in this sense, a core computer-science view of the issues.)

Research papers

Website(s): http://www.cs.man.ac.uk/~sattler/teaching/cs636.html

***Synopsis**:*

Part 1. Data Warehousing:

- Introduction to Data Warehousing: Heterogeneous information; the integration problem; the Warehouse Architecture; Data Warehousing; Warehouse DBMS.
- Aggregations: SQL and aggregations; aggregation functions; grouping.
- Data Warehouse Models and OLAP Operations: Decision support; Data Marts; OLAP vs OLTP; the Multi-Dimensional data model; Dimensional Modelling; ROLAP vs MOLAP; Star and snowflake schemas; the MOLAP cube; roll-up, slicing, and pivoting.
- Some Issues in Data Warehouse Design: monitoring; wrappers; integration; data cleaning; data loading; materialised views; warehouse maintenance; OLAP servers; metadata.

Part II. Data mining:

- Introducing Data Mining: Why data mining?; What is data mining?; A View of the KDD Process; Problems and Techniques; Data Mining Applications; Prospects for the Technology.
- The CRISP-DM Methodology: Approach; Objectives; Documents; Structure; Binding to Contexts; Phases, Task, Outputs.
- Data Mining Inputs and Outputs: Concepts, Instances, Attributes; Kinds of Learning; Providing Examples; Kinds of Attributes; Preparing Inputs. Knowledge Representations; Decision Tables and Decision Trees; Classification Rules; Association Rules; Regression Trees and Model Trees; Instance-Level Representations.
- Data Mining Algorithms: One-R; Naïve Bayes Classifier; Decision Trees; Decision Rules; Association Rules; Regression; K-Nearest Neighbour Classifiers.
- Evaluating Data Mining Results: Issues in Evaluation; Training and Testing Principles; Error Measures, Holdout, Cross Validation; Comparing Algorithms; Taking Costs into Account; Trade-Offs in the Confusion Matrix.

Assessment: One 2-hours midterm exam (30%); Assignments (15%); Seminars (15%); 2-hours Final Exam (40%)

……..……

750771, Multimedia Systems Technology

3 hours per week, 3 credit hours, prerequisite: **none**

Teaching Method: 37 hours Lectures (2-3 hours per week), 8 hours Seminars (1 per 2 weeks)

Aims: The modules aims to introduce topics in digital audio and video, different interchange formats, multimedia hardware, multimedia software, multimedia communications, videoconference, and multimedia systems.

Learning Outcomes:

On completion of this module, the student should be able to:

- Understand different multimedia hardware and software.
- Understand some real-time transfer and control protocol.
- Design audio and video applications.
- Work with animation.

Textbooks and Supporting Materials:

1- Chan-Hwa Wu and J. David Irvin, Emerging Multimedia Computer Communication Technologies, Prentice Hall, 1998

Synopsis*:*

1. Introduction: What is multimedia; Multimedia systems; Quality of service; Synchronization and orchestration; Standards; Convergence; Value chain;
2. Hardware: Multimedia computers; Video and graphics; Audio; Telephone, videoconference, and networks; CD and DVD; USB and FireWire; Processors; Video for Windows, DirectX, and ActiveMovie.
3. Software: Introduction; Browser based software architecture; Distributed software; Servers; Network Terminals.
4. Audio and Video I: Introduction; Digital audio; Psychoacoustics; Digital presentation of sound Digital images; JPEG.
5. Audio and Video II: Video signal; Camera sensors; Colors; Color television; Equipment; Compression systems; Basics of video compression; Methods; Algorithms.
6. Interchange Formats: Introduction; Application areas; Requirements; Track and object model Real-time transfer; Different transfer formats; Comparison.
7. Authoring Tools: Introduction; Production process; Tools; Barriers; Development areas
8. Communications: QoS; ATM; QoS implementations; Integrated Services; Differentiated Services.
9. Multicast: Introduction; Group control; Routing; Real-time transfer and control protocols; Resource reservation; Session control; MBone .
10. Video Conference: Introduction; Standards; Products; Internet telephony; CTI (Computer Telephony Integration).
11. Access Networks: Introduction; Cable television; Digital subscriber lines; UMTS; Digital television; Conclusions.

Assessment: One 2-hours midterm exam (30%); Assignments (20%); Seminars (10%); 2-hours Final Exam (40%)

……..……

750772, Multimedia and the Web

3 hours per week, 3 credit hours, prerequisite: **none**

Teaching Method: 37 hours Lectures (2-3 hours per week), 8 hours Seminars (1 per 2 weeks)

Aims: This module introduces the design and development of effective multimedia systems and develops their awareness of current trends and developments in multimedia applications and systems. Also it enables students to evaluate the systems' features and effectiveness.

Learning Outcomes:

On completion of this module, the student should be able to:

- Understand many multimedia systems.
- Design and develop effective multimedia systems.
- Be aware of current trends in multimedia applications and systems.

- Evaluate the multimedia systems' features and effectiveness.

Textbooks and Supporting Materials:

1- Steve Grosvenor, The Flash Anthology: Cool Effects and Practical ActionScriptBAuthor: 1st ed. 2004.

Synopsis*:*

1. Introduction to Multimedia: This chapter describes what multimedia is and how multimedia can be handled by web browsers.
2. Sound Formats: This chapter describes the most common - or popular - sound formats.
3. Video Formats: This chapter describes the most common - or popular - video formats.
4. Browser Sounds: This chapter describes how to play sounds from a web page.
5. Browser Videos: This chapter describes how to play videos from a web page.
6. Windows Media Formats: This chapter describes the new Windows Media formats.
7. Object Intro: This chapter describes the object element.
8. Object QuickTime: This chapter describes how to play QuickTime movies with the object element.
9. Object RealMedia: This chapter describes how to play Real Audio and Real Video with the object element.
10. Tag Reference: The HTML multimedia tag reference.
11. Player Reference: The Windows Media Player reference.

Assessment: One 2-hours midterm exam (30%); Assignments (20%); Seminars (10%); 2-hours Final Exam (40%)

……..……

750781, Formal Methods in Software Engineering

3 hours per week, 3 credit hours, prerequisite: **none**

Teaching Method: 37 hours Lectures (2-3 hours per week) + 8 hours Semonairs (1 per 2 weeks)

Aims: This module is to introduce basic concepts of formal methods and their practical applications in software engineering. Students will learn why and how formal methods should be used in the entire software development process for delivering a product of quality assurance. The following topics will be discussed: formal methods based software life-cycle models; languages for software system specification; modeling and abstraction of software systems; Software architectures, analysis and verification of system properties; system refinement and program transformation; formal semantics, program specification and verification.

Learning Outcomes:

On completion of this module, the student should:

- Understand the concepts of formal methods and their usage in software engineering.
- Have knowledge on languages for software system specifications.
- Be able to use the formal methods in software development process.

Textbooks and Supporting Materials:

I. Textbooks:

1- Formal Methods for Software Architectures: Third International School on Formal Methods for the Design of Computer, Communication and Software Systems: Software Architectures, SFM 2003, Bertinoro, Italy, September 22-27, 2003, Advanced Lectures.
2- J. B. Wordsworth, Software Development with Z, Addison Wesley,
3- Michael G. Hinchey & Jonathan P. Bowen, Applications of Formal Methods, Prentice Hall, 1995.
4- Matt Kaufmann, Panagiotis Manolios, and J Strother Moore, Computer-Aided Reasoning: An Approach, Kluwer Academic Publishers, June, 2000. (ISBN: 0-7923-7744-3)
5- Doron A. Peled, Software Reliability Methods. Springer-Verlag, 2001.
6- Johann M. Schumann, Automated Theorem Proving in Software Engineering, Springer-Verlag, 2001.

II. Selected Research papers (Formal Methods Resources Websites)
III. Z software development tools

Synopsis*:*
1- Introduction: What are Formal methods?; Why Formal methods?; Formal methods and Software Process.
2- Requirements, Models, Formal Specifications.
3- Formal Analysis and Verification.
4- Software Development in Z and in ObjectZ.
5- Formal Semantics.
6- Refinement and Program Transformation.
7- Program Specification and Verification.
8- Formal methods for Software Architectures.

Assessment: One 2-hours midterm exam (30%); Assignments (15%); Seminars (15%); 2-hours Final Exam (40%)

……..……

750782, Software Process

3 hours per week, 3 credit hours, prerequisite: **none**

Teaching Method: 37 hours Lectures (2-3 hours per week), 8 hours Seminars (1 per 2 weeks)

Aims: This module aims to provide students with knowledge about: description of commonly used software life cycle process models and the content of institutional process standards; definition, implementation, measurement, management, change and improvement of software process; and use of a defined process to perform the technical and managerial activities needed for software development and maintenance.

***Learning Outcomes*:**
On completion of this module, the student should:
- Have knowledge on commonly used software life cycle process models.
- Understand the standards of software process
- Be able develop software projects
- Be able to use a defined process to maintain software.

Textbooks and Supporting Materials:
1. R. S. Pressman. Software Engineering: a Practitioner's Approach, 6th edition, McGraw-Hill, 2005 .
2**.** Daniel Galin, Software Quality Assurance**,** ISBN 0 201 70945 7,

Research papers:
Website(s): http://www.mhhe.com/pressman
http://www.booksites.net/download/galin/download.htm

Synopsis*:*
Part 1- Software Process:
1- Introduction: Motivation, Software, process, Software Process, Model = Methodology + Structures.
2- Conventional Methodologies: All known software process (waterfall, etc.)
3- An Agile Process
4- Software Process Fundamentals: *Requirements:* Modeling, evolution, formalism, enactment, programming, environment; *Concepts:* Activity, Agent, Role, Event, Constraint, Product, Coordination; *Paradigm:* Programming Model, Active database, Modeling language and IA, Network, Hybrid; *Issues:* Integration, evolution, Coherence, Coherence, Environment; *Methodologies and Structures:* Individual Models, Collective Model, Methodology, structures**.**
5- Specific Software Process: Methodology**;** Structures; Generic Software Process.
6- Generic Software Process: Methodology; Structures; Generic Software Process.

Part 2-Software Quality:
7- Quality Management
8- Product Measurement and Metrics
9- Process and Project Measurement and Metrics

Assessment: One 2-hours midterm exam (30%); Assignments (10%); Seminars (20%); 2-hours Final Exam (40%)

……..……

750783, Software Maintenance

3 hours per week, 3 credit hours, prerequisite: **none**

Teaching Method: 37 hours Lectures (2-3 hours per week), 8 hours Seminars (1 per 2 weeks)

Aims: The Software Maintenance is a most large and expensive task (75%) in the life cycle of software. The practical importance of this task has attracted great attention these last years, at both academic and industrial levels. This module aims to present theoretical and a practical techniques, tools, and methodologies that help software developers to achieve this unavoidable task with a maximum of success.

Learning Outcomes:
On completion of this module, the student should:
- Have knowledge on theoretical and a practical techniques, tools, and methodologies of software maintenance.

- Understand the importance of software maintenance.
- Be able to use Reverse Engineering Tools.
- Be able to design error free software.

Textbooks and Supporting Materials:

1- R. S. Arnold, Software Reengineering, IEEE , 1993
2- T. M. Pigoski, Practical Software Maintenance: Best Practices for Managing Your Software Investment, Wily, 1997.

Research Papers:

1- J. Estublier, S. Ghoul. Preliminary Experience with a Configuration Control System for Modular Programs, ACM SIGSOFT (USA), Vol.9, No.3, May 84. pp. 149 - 156.
2- R. Conardi and B. Westfechted, Versions Models for Software Configuration Management, ACM Computing Surveys, Vol.30, Issue 2, 1998.
3- T. Khammaci, ZE. Bouras, S. Ghoul, Program Slicing: Precise Shops Extraction Approaches, Hand book of Software Engineering and Knowledge Engineering, Vol. 1, Fundamentals, S. K. Chang (editor), Word Scientific Publishing, 2001.
4- MS Bendelloul, S. Ghoul, T. Khammaci, An object-based Decomposition for Assistance to Software Maintenance, Proc. of the 5th Magrrebian Conf. on Software Engineering and Artificial Intelligence, MCSEAI'98, Tunis, Tunisia, 1998.
5- MS Bendelloul, S.Ghoul, An Object System for Software Maintenance, Proc. of the 4th African Conference on Computer Science, CARI'98, Dakar, Senegal, 1998.

Website(s): www.suite101.com/subjectheadings/contents.cfm/671

Synopsis*:*

1- Introduction: Software maintenance methods; Software maintenance tools.
2- Configurations Management: Configuration Database; Configurations Dependencies; Evolution Constraints, Automatic Evolution; Applications: Automatic Software Versions Generation
3- Restructuring tools: Error avoidance tools; Architectural tools; Application: Pretty-Print.
4- Reverse Engineering Tools: Reverse Engineering Tools [Knowledge-based]: Errors Diagnosis, application; Reverse Engineering Tools [Static Analysis]: Dependency Graph, Slicing, Shopping; Reverse Engineering Tools [Static Analysis]: Call Shopping, Application: Modification effects and damage assessment; Reverse Engineering Tools [Object-oriented Programming]: Behavior-Based decomposition; Object behavior and maintenance, Application: Automatic Program Understanding.

Assessment: One 2-hours midterm exam (30%); Assignments (10%); Seminars (20%); 2-hours Final Exam (40%)

……..……

750799, Thesis

9 credit hours

General Descriptions:

After passing the compulsory and elective modules, the student can select one of the projects announced by the Computer Science department under the supervision of a member of staff. The project consists of a research work on which the student works over a period of two semesters that can be extended to three semesters.

Aims: The aims for the project work are:
1- To manage and execute a substantial project in a limited time.
2- To identify and learn whatever new skills are needed to complete the project.
3- To apply design and engineering skills in the accomplishment of a single task. In this context the skills mentioned may be in the general area of design and engineering in its broadest sense, or may be very specifically related to particular tools.

Learning Outcomes:
On completion of this module, a student should have
1. Planned, executed and completed a significant design and implementation within the time.
2. Used the project supervisor appropriately as project consultant or customer.
3. Given a seminar which justifies the project.
4. Documented the project in a final dissertation.
5. Learned an approach to scientific research.

Textbook:
C. W. Dawson, The Essence of Computing Projects, A Student's Guide. ISBN 0-13-021972-X. Prentice Hall 2000.

Assessment:
The formal project deliverables are two seminars and a written dissertation. Examination committee will assess the quality of the theis document and students' defense.

• **Seminars**
The seminar is a formal presentation on the project given to members of staff, together with any other students who wish to attend. The duration is about 30 minutes, which includes reasonable time for questions.

Students are expected to give the first seminar at the end of first semester. The talk should cover a description of the general problem and the background. The second seminar will take place at the end of second semester. In this seminar, the general method of solution and the main results of the project are demonstrated.
The seminars are to give students experience in communicating their work to others in a formal manner.

• **Dissertation**
The dissertation is a formal written document on the project. The dissertation must follow the following set of standards, to facilitate its inclusion in the library and its usefulness for subsequent readers.
A) Here is a suggested structure for the dissertation. Some projects may be rather different from others, and therefore have good reasons for not following these suggestions exactly. Supervisor guidance should anyway be sought!
   1. Introduction (1st chapter). What is the overall aim of the project? Why is it worth doing? Who will benefit from it? If the overall aim can be split into a number of subgoals, this is a possible place to do it. Finish with a chapter by chapter overview of the rest of the dissertation.

2. Background (2nd chapter). Analyze the background to the project. This should mention any previous work, here or elsewhere, and explain its relevance to the project. This could be an appropriate place to justify the choice of platform/software etc. used in the project.
3. Description of the student's own work: Design and Implementation (a chapter each). The structure of these chapters may reflect the project lifecycle, but do not write a diary of progress. The design should be clearly described and justified. Supporting diagrams should be used where appropriate and helpful. Keep your design description fairly high level. When describing implementation, confine yourself to the important, difficult, or interesting bits. Do not include large chunks of code. Figures may well be useful.
4. Results (1 chapter). What is the resulting system like to use. Include screen shots as appropriate.
5. Testing and Evaluation (1 chapter). What testing was done? How confident are you that everything works correctly, and what evidences can you produce to support this claim? Have you evaluated the system against its aims? How did you make this evaluation?
6. Conclusions (last chapter). What conclusions can you draw from the whole project? This should include a clear statement of what has been achieved overall, and will normally continue by suggesting areas of further related work which could be done.

B) The dissertation forms the basis of an independent assessment of the project and therefore has greater effect on the final project mark.

C) The dissertation must be on paper of A4 size (210 x 297 mm). Only one side of paper should be used. The dissertation must be produced using word processing facilities.
The body of the dissertation should be suitably divided into chapters and sections. Chapters, sections, pages, figures and appendices should all be numbered. Each chapter should start on a new page.
The body of the dissertation should be preceded by a temporary title page, an abstract and a list of contents, and it should be followed by the references and then any appendices.

References to other published work should follow the conventions used in giving references in published work. e.g.:

[1] P. J. Denning, Human Error and the Search for Blame, *Communications of the ACM* 33(1): pp 6-7, January 1990.

The abstract page must give the title, author, and supervisor, as well as an abstract of the project.

For further details, see the regulations of graduate studies set by the Deanship of postgraduate studies and scientific research in the university.