MSc Syllabus: Advanced Computer Science Course Units

2004/2005

The University of Manchester, School of Computer Science

Oxford Road, Manchester M13 9PL, U.K.

Version of September 14, 2004
URL: http://www.cs.man.ac.uk/Study_subweb/Postgrad/ACS-CS/webpages/syllabus/acs/

This document describes the Advanced Computer Science course units and the aims and learning outcomes of the MSc in Advanced Computer Science for the current session.

The course units change yearly in order to keep up with new developments in the subject. Most, if not all, of the units listed below will be available in the forthcoming year. However, various factors may cause a course unit to be withdrawn (including availability of staff and demand from students) and so there is no guarantee that any particular course unit will be available, though as a School we try to ensure that all are. The number of students on a course unit is normally limited to 50, but, because of facilities or teaching methods, other limits may be imposed on individual units.

Some of the course units are specifically designed for particular MSc programmes. They may however be available for students on other MSc programmes if facilities and other arrangements allow. If you wish to take such course units you will need the approval of both your programme director and the programme director responsible for the course unit. This applies in particular to those marked EIS (Electronic Instrumentation Systems), to those for the MSc in Computational Science and Engineering (CS607, 608, 609), to those for the MSc in Advanced Computer Science with ICT Management (CS851, 852), and those supplied by the PEVE Unit via E-learning. Other course units may have restrictions indicated in the syllabus document.

Contents

1 Programme Aims and Learning Outcomes

Programme Aims

The educational aims of the Advanced Computer Science Masters are to:

- Produce the highest quality of computing professionals and researchers across a broad range of Computer Science
- Provide a vehicle for dissemination of leading edge knowledge and skills, focussing on the research strengths of a large school covering most major topics in Computer Science and its applications
- Offer the opportunity to focus on one of a range of specialisations.

Programme Learning Outcomes

A. Knowledge and understanding

A1. Knowledge of a range of advanced topics, both at the applied level and the research level, beyond undergraduate level and at the forefront of research.

A2. Leading-edge technologies in one or more of: advanced computer architectures, formal foundations of Computer Science, software engineering, advanced applications, artificial intelligence.

B. Intellectual Skills. The ability to:

B1. develop original ideas in a research context.

B2. use methodologies for development of computational systems at an advanced level.

B3. perform problem solving in academic and industrial environments.

C. Practical Skills. The ability to:

C1. develop applications to satisfy given requirements.

C2. organise and pursue a scientific or industrial research project.

C3. use, manipulate and create large computational systems.

C4. perform independent information acquisition and management.

D. Transferable Skills. The ability to:

D1. work effectively as a team member.

D2. prepare and present seminars to a professional standard.

D3. write theses and reports to a professional standard, equivalent in presentational qualities to that of publishable papers.

D4. perform independent and efficient time management.

D5. use a full range of IT skills and display a mature computer literacy.

Advanced Course Unit Descriptions

2 CS600: Introduction to Advanced Computer Science

Level:	MSc
Credit Rating:	None
Degrees:	ACS/CS/CompSci
Pre-requisites:	None
Pre-course work:	None
Taught week:	40 hours: introductory seminars and activities
Post-course work:	None
Assessment:	None
Lecturer(s):	Programme Director, Tutor and Lecturers.

Introduction

This is an important and enjoyable overview of the course and of advanced topics in computer science. For each taught course unit, there is an introductory talk given by one of the lecturers of the course unit. This is an opportunity to learn what each topic is about, what problems it tackles, what skills and knowledge are required and learned. This also provides a forum for discussing each topic with an expert in the area. The student is expected to attend all the introductory talks, viewing this not simply as an opportunity to choose a selection of course units, but also as an opportunity to broaden knowledge and see what are the concerns of other topics across the range of Computer Science. The course unit also covers material on the structure and expectations of the course, and an introduction to the facilities of the School, the Graduate School and the University.

Aims

This course unit has three aims.

- To provide an overview of the course itself, its aims and structure and an introduction to the School and its equipment and to the University,
- To provide a broad overview of the major issues, themes and topics in advanced computer science,
- To provide introductions to each of the optional course units, given by the staff who will teach them. These will assist students in making their selection of six of the course units, giving an explanation of the issues, knowledge and skills dealt with in each course unit.

Learning Outcomes

After completion of the course unit, the student will understand the structure of the Advanced MSc course, what is expected of the student and procedures for the course.

Through the introductory talks for the course units, the student will have gained a wide overview of advanced topics in computer science and will be able to select course units for further study with an understanding of what each topic is, its applications and relationship with other areas, and what skills and knowledge is to be gained through each course unit. (A)

Assessment of learning outcomes

There is no assessment for this course unit.

Reading list and supporting material

Lecturers will supply material about each of the course units.

Special resources needed to complete the course unit

No special resources.

Detailed syllabus

- Introduction to the course, the School and the University.
- Major issues in advanced computer science.
- Introductions to individual course units.

3 CS602: Grid Computing and eScience

Level:	MSc
Credit Rating:	15
Degrees:	ACS/CompSci
Pre-requisites:	None
Pre-course work:	40 hours: 30% introductory lab, 70% preparatory reading
Taught week:	40 hours: 50% lectures, 50% supervised lab
Post-course work:	40 hours: 100% Mini Project
Assessment:	35% exam, 30% coursework 35% MiniProject
Lecturers:	Dr.John Brooke, Dr Stephen Pickles, Dr Jon MacLaren, Mr Donal Fellows
Limit on numbers:	50 participants

Introduction

Grid computing and eScience are two major areas of growth in the field of distributed systems. The Grid concept refers to the virtualisation of computing resource in the sense that end-users should have the illusion of using a single source of "computing power" without knowing the locality of the computation. Examples of this virtualisation are the use of digital certificates to access systems on behalf of the user, third party file transfer between machines authenticated via certificates, client tools for workflow composition with the workflow being consigned by agents such as brokers. There is a growing movement of convergence with the Web services community and this is attracting the interest of major companies such as IBM. HP, SUN., SGI, who see their future business increasingly involving the provision of an infrastructure where computing services are traded between providers rather than individual groups within an organistation having their "own" machines.The course will introduce the concepts and develop lab exercises based on job submission and monitoring on a local Grid.The course tutors are all active in the Global Grid Forum which is becoming the body for determining Grid standards, thus this course will be informed by the very latest developments in this highly dynamic field. EScience is allied to the Grid concept, it refers to new methods of utilising Grid and other forms of distributed computation with a particular emphasis on collaborative working by geographically distributed teams. Much academic and industrial research and development is increasingly utilising the eScience model, which also goes under the title of "Cyberinfrastructure" (the latter term being used in the US).

Aims

This course unit aims to:

1. explain the concept of eScience and its importance in future problem solving IT infrastructure.
2. explain the concept of Grid computing and its relation to eScience,
3. familarise students with the key abstractions underpinning the Grid concept,
4. outline current Grid solutions and how they are intended to evolve,
5. give a more in-depth view of a widely used Grid middleware system UNICORE,
6. give lab sessions in running Grid computing jobs using the UNICORE GUI based job composition and submission method,
7. provide a mini-project to explore some particular aspects of Grid computing, e.g. resource discovery, application plugins, workflow composition.

Learning Outcomes

A student successfully completing this course unit should:

1) Have an understanding of the concepts of Grid computing and eScience and why they have assumed such current prominence. In particular to have an understanding of the importance of standards and protocols in Grid computing (A),

2) Understand the architecuture of the UNICORE middleware and how this relates to the emerging Open Grid Services Architecture proposals and standards (A,B)
3) Be able to utilise UNICORE to submit both simple and multistage computing jobs onto a local Grid (A,B,C),
4) Be able to explore via UNICORE a particular aspect of Grid computing, for example in obtaining information about resources on wide area Grids, extending the UNICORE system via an application specific plugin, investigation interoperability with other Grid systems (e.g. Globus) (A,C).

Assessment of learning outcomes

Learning outcomes (1) and (2) are assessed by examination, learning outcome (3) is assessed by laboratory reports, learning outcome (4) is assessed by mini-project.

Contribution to programme learning

A1, A2, B2, B3, C1, C3, D1, D5.

Reading list and supporting material

There is a CS602 web page[1]. Follow links from Documents for current session.

Grid computing is so dynamic that most books are either not written or are out of data. The original and most influential book is

– I. Foster and C. Kesselman eds., *The Grid: Blueprint for a new computing infrastucture* Morgan Kaufmann, San Francisco, 1998.

DON'T buy the first edition, the second edition should be out in mid 2003. The first edition is worth reading but is considerably out of date.

The best way to get information is to search the Web with the keywords Grid, eScience, Unicore, Globus. A useful reference site for the course unit is here[2].

You may wonder why UNICORE is taught in preference to Globus. It has a more compact architecture which makes it more suitable for this level of study and it is closer in structure to the Open Grid Services Architecture which will be the standard for future Grid computing. UNICORE can be used to run jobs on a Globus Grid so there are no restrictions implied by the choice of this system and the aim of the course is to present the principles underlying the most widely used Grid middleware systems.

Special resources needed to complete the course unit

It should be possible to install the course software anywhere, but attention will need to be paid to security since Grid access is a very powerful tool and course participants accessing from outside the school computing resources may be required to sign forms to obtain certificates from the Certificate Authority.

Detailed syllabus

First we define the lecture syllabus. Each lecture will be of one hour duration. Each topic will have 3 lectures devoted to it making 12 lectures in all. The rest of the time will be devoted to the laboratory classes listed below.

[1] http://www.esnw.ac.uk/
[2] http://www.grid-interoperability.org

1. The metacomputing problem: forerunner to the Grid. Exploring the convergence of exploitation of high speed networks, exploitation of architectural affinity, work on coupled multiphysics problems, e.g. Climate Models importance of locality requirements to minimise flow of data across wide area networks.
2. Grid computing: a persistent metacomputing environment. Digital certificates as a persistent and scalable form of authorisation, Virtualisation of resources, hiding of complexity of metacomputing environment from user.
3. Role of middleware in Grid computing. Neccesity for abstractions in a heterogeneous environment, differing OS's, resource management systems, programming languages. Interoperability achieved via tiered middleware architectures.
4. Abstract modelling approach to middleware problem - UNICORE Concept of an Abstract Job Object and its relation to workflow composition and enactment. Concept of Incarnation from abstract resource space to concrete resource space. Vertical integration in UNICORE, difference between a tiered and a layered model.

The laboratory sessions will cover the rest of the time:

- Use of UNICORE GUI to compose a Grid workflow
- Use of UNICORE Resource Broker to locale a suitable machine or machines on a local Grid
- Submission and monitoring of the job via the UNICORE client.
- Dealing with job termination and tidy up.
- Architecture extension: installing a simple client plugin for UNICORE

4 CS603: High Performance Computing in Science and Engineering

Level:	MSc
Credit Rating:	15
Degrees:	Advanced Computer Science, Computational Science (possibly CS)
Pre-requisites:	None
Pre-course work:	40 hours: 30% introductory lab, 70% preparatory reading
Taught week:	40 hours: 50% lectures, 50% supervised lab
Post-course work:	40 hours: 100% mini-project
Assessment:	35% exam, 30% lab, 35% mini-project
Lecturer(s):	Dr T L Freeman, Prof J R Gurd, Mr G D Riley
Limit on numbers:	50 participants

Introduction

Today's highest performance computers embody substantial amounts of parallel hardware, to the extent that the latest generation of machines harness the power of thousands of cooperating processors. The programming of such highly parallel hardware has proved to be difficult: progress has been slow and achieved mostly by "trial-and-error". Convergence between the competing technologies has taken unusually long and the HPC market remains highly volatile.

Aims

This course unit studies the base technologies for HPC and allows "hands-on" experience of a state-of-the-art parallel supercomputer to be gained. The course unit explores, through a combination of directed reading, lectures, group-based laboratories and group-based mini-projects, a framework for the development, analysis and performance tuning of parallel algorithms for the solution of numerical problems.

Learning Outcomes

A student completing this course unit should:

1) have an understanding of the different levels of abstraction in HPC Modelling and of different parallel programming models; (A)
2) have an understanding of parallel performance overheads and of techniques for reducing them; (A)
3) be able to implement a moderately complicated application in a parallel language; (B and C)
4) have an understanding of some parallel numerical algorithms; (A)
5) be able to work effectively as a member of a group to develop parallel applications. (D)

Assessment of learning outcomes

Learning outcomes (1) and (2) are assessed by examination, in the laboratory and via the mini-project, learning outcomes (3) and (5) are assessed in the laboratory and via the mini-project, and learning outcome (4) is assessed by examination.

Contribution to programme learning

A1, A2, B2, B3, C1 and C3.

Reading list and supporting material

Culler, D E and Singh, J.P.with Gupta, A., Parallel Computer Architecture: A Hardware/Software Approach, Morgan Kaufmann Publishers, 1999.

Foster, I., Designing and Building Parallel Programs. Addison-Wesley, 1995.

Hennessy, J.L. and Patterson, D.A., Computer Architecture A Quantitative Approach. Morgan Kaufmann, 1996.

Special resources needed to complete the course unit

Access to parallel computers situated in the School.

Detailed syllabus

Introduction to HPC Why is HPC important in Science and Engineering? Introduction to Parallel Computers and Computational Overheads.

Levels of Abstraction, Models of Computation and Parallel Overheads Levels of Abstraction, Multiple Program Counters in Hardware; Multi-Thread Models, with Primary Sources of Overhead; Parallel Languages and Compilers; Task-Parallel versus Data-Parallel Programming Models; Further Sources of Overhead; Experimentation and Presentation of Results; Memory Architecture and Memory Access Times and Associated Sources of Overhead; Multi-Process Execution Model; Performance Tuning via Overhead Reduction; Task Scheduling; Data Partitioning and its Effect on Performance.

Restructuring for Parallel Performance Parallelising Compilers; Loop Transformations; Data Transformations; Dependence Analysis; Compiler Strategies.

Parallel Algorithms Examples of Parallel Algorithms: Cyclic Reduction; Iterative Algorithms (Jacobi, Gauss-Seidel and Red-Black Orderings); Divide-and-Conquer Algorithms, Adaptive Quadrature, Correct Termination.

Resumé Review of the course material and the unifying theme of levels of abstraction.

5 CS605: Visualisation for HPC

Level:	MSc
Credit Rating:	15
Degrees:	ACS/CompSci
Pre-requisites:	None
Pre-course work:	40 hours
Taught week:	40 hours
Post-course work:	40 hours
Assessment:	33% exam, 67% practical exercise
Lecturer:	Terry Hewitt and Dr. N.W. John
Limit on numbers:	50 participants

Aims

The quantities of data produced by simulations on supercomputers of physical, natural and theoretical problems are frequently so large that graphical representations offer the only viable way to assimilate them.

Often, the simulation models themselves are complex, involving large numbers of independent and dependent variables whose relationships need to be understood. For example, in climate modelling, we may wish to explore how temperatures, water vapour content, pressure, wind directions and velocities vary within a 3D region, over time. The process of visualisation is therefore concerned with ways to represent the data, and tools for interactive exploration of multi-dimensional, multi-variate models. An active research area is to find ways to link this visualisation process with interactive control of the simulations themselves, opening up completely new possibilities for interactive exploration and understanding of complex phenomena. Recently, a number of visualisation systems have emerged, which provide a framework for this kind of model exploration.

The aims of this course unit are to provide a practical introduction to computer-aided visualistaion of such complex data, using systems to interactively explore models of data. The course unit combines lectures and background reading, together with laboratory exercises and mini-projects using the application visualisation system (AVS).

Learning Outcomes

A student completing this course unit should:

Understand and be capable of using visualisation tools in key areas which contribute to successful visualisation, including understanding the dimensionality of data, reference models for data manipulation and mapping, display techniques and algorithms (including parallel algorithms), use of colour, visual perception, user interfaces (including cooperative working and virtual reality), use of animation, and visualisation systems. (A, B and C)

Assessment of learning outcomes

The practical skills of using visualisation tools are assessed in the practical exercise, which consists of a mini-project undertaken in groups. The examination tests the understanding and knowledge described above.

Contribution to programme learning

A1, A2, B3, C1, C3 and D1.

Reading list and supporting material

There is no recommended textbook for this course, but handouts and lists of suggested papers will be given to students. The following two books are suitable background reading.

Rosenbaum, L. et al. (ed.), Scientific visualization, advances and challenges. IEEE Society Press, Academic Press; and Scott Whitman, Multi processor methods for computer graphics rendering. Jones and Bartlett, ISBN:0-86720-229.

Special resources needed to complete the course unit

The AVS visualisation software is used on this course unit.

Detailed syllabus

Lectures

- Intro to the course unit
- Overview of visualization.
- Information into pictures.
- 2D Visualization.
- Volume visualization.
- Reference models.
- Flow visualization.
- Graphics hardware and software.
- Interpolation and approximation.
- Visual perception and colour.
- Interaction basics.
- Modes of interactions.
- Parallel architectures.
- Parallel visualization.
- Multidimensional data.
- Remote visualization.
- Visualization systems.
- Data management.

Laboratory Sessions

- AVS training, NW editor, geometry viewer, volume visualization, 3D scalar, medical imaging.
- Climate model 1.
- Climate model 2.
- CFD working demo.
- Cooperative working demo.1.
- CFD Double glazing 2. Make a Video.
- Implementing marching cubes 1.
- Implementing marching cubes 2.
- Implementing marching cubes 3.

Mini Projects

Mini-projects will be done in groups of 3 using the CGU HP/SGI or the MSc equipment.

6 CS607: Introduction to Computational Science

Level:	MSc
Credit Rating:	15
Degrees:	Computational Science (ACS if qualified)
Pre-requisites:	None
Pre-course work:	40 hours: 30% introductory lab, 70% preparatory reading
Taught week:	40 hours: 50% lectures, 50% supervised lab
Post-course work:	40 hours: 100% mini-project
Assessment:	35% exam, 30% lab, 35% mini-project
Lecturer(s):	Dr T L Freeman, Professor J R Gurd, Dr M Mihajlovic, Mr G D Riley
Limit on numbers:	50 participants

Introduction

Aims

Modern Science and Engineering have become increasingly dependent on large numerical simulation to aid progress in research, development and design. It is difficult to think of a large-scale Science or Engineering project that does not rely on some aspect of Computational Science. The aim of this course unit is to provide an introduction to the range of issues (algorithmic, software and hardware) that need to be addressed to derive efficient and adaptable numerical solutions of some simple PDEs that model physical problems.

Learning Outcomes

A student completing this course unit should:

1) have an understanding of the execution cycle of a numerical code that simulates a simple PDE; (A)
2) have an understanding of the factors in the execution cycle that affect performance on a sequential machine;
3) have an understanding of the benefits of abstraction in program design. (D)

Assessment of learning outcomes

Learning outcomes (1) and (3) are assessed in the laboratory and via the mini-project, learning outcomes (2) is assessed by examination, in the laboratory and via the mini-project.

Contribution to programme learning

A1, A2, B2, C1 and C3.

Reading list and supporting material

M.T.Heath, Scientific Computing: An Introductory Survey, 2nd edition, McGraw-Hill, New York, 2002. ISBN 0-07-239910-4
D.Besset, Object-Oriented Implementation of Selected Numerical Methods: An Introduction with Java and Smalltalk, Morgan Kaufmann Publishers, 2000. ISBN 1558606793
D.Flanagan, Java in a Nutshell: a Desktop Quick Reference, 4th edition, O'Reilly, 2002. ISBN 0-596-00283-1

Special resources needed to complete the course unit

Detailed syllabus

Simple examples Simple motivating examples (with simple boundary conditions):

- elliptic PDE (Laplace),
- parabolic PDE (heat conduction).

Discrete formulations Linear equation solution methods (direct vs. iterative). Scaling with problem size (dependence of error on mesh size).

Difference Equations Difference equations into code. Difference equations, systems of linear algebraic equations, linear equation solution techniques (both direct and iterative), Java code, visualisation of solutions. Object-orientation for high level "plug'n'play".

Details of solvers Solver internals and introduction to system architecture. Only selected effects to be explained; stride access to memory, JIT optimisations.

Component-based Analysis and Design Component-based Analysis and Design of the example problems at all levels of abstraction: Physics; Finite Difference Equations; Algebraic Equations; Matrix element level.

Resumé Search for best (most efficient) solution to the simple examples and illustration that a performance problem remains. Consider implications of increased problem complexity for performance; more complex domains, more complex PDEs.

7 CS608: Fundamentals of High Performance Execution

Level:	MSc
Credit Rating:	15
Degrees:	Computational Science (ACS and others if qualified)
Pre-requisites:	CS607
Pre-course work:	40 hours: 30% introductory lab, 70% preparatory reading
Taught week:	40 hours: 50% lectures, 50% supervised lab
Post-course work:	40 hours: 100% mini-project
Assessment:	35% exam, 30% lab, 35% mini-project
Lecturer(s):	Professor J R Gurd, Mr G D Riley,
Limit on numbers:	50 participants

Introduction

Aims

To introduce to non-Computer Science graduates the Fundamentals of System Architecture. Thus to enable them to understand the execution characteristics of scientific simulation code.

Learning Outcomes

A student completing this course unit should:

1) have an understanding of the essentials of serial program execution cycle: source code $\longrightarrow$ object code $\longrightarrow$ run-time code $\longrightarrow$ hardware; (A)
2) have an understanding of the effects of components of execution cycle on performance;
3) have an understanding of the limitations of abstraction, in particular in terms of program performance. (D)

Assessment of learning outcomes

Learning outcomes (1) and (2) are assessed by examination, in the laboratory and via the mini-project, learning outcomes (3) is assessed in the laboratory and via the mini-project.

Contribution to programme learning

A1, A2, B2, C1 and C3.

Reading list and supporting material

J.L.Hennessy and D.A.Patterson, Computer Architecture: A Quantitative Approach, Morgan Kaufmann, 1996.

K.Dowd and C.Severance, High Performance Computing, O'Reilly, 1998.

Special resources needed to complete the course unit

Detailed syllabus

High level view of source code to run-time code transformation Fundamentals of Compilation; Optimisation; Interpretation; Libraries.

High level view of run-time code to hardware transformation Simple architectural model - one instruction at a time (issued at a constant rate), constant memory access time, performance implications.

Lower level view of source code to run-time code transformation Formal description of program (Abstract Syntax Tree, Call Graph, Dependence Graph); Compiler Optimisations (Simple Data Dependences, Transformations, etc.); JIT Compilation.

Lower level view of run-time code to hardware transformation More realistic (complex) architectural model - memory hierarchy (effects of cache), multi-way instruction issue, etc, performance implications. Evaluation of real performance effects on real hardware.

Resumé Measure of performance refined from flops (floating point operations per second) to ips (instructions per second) to actual hardware performance; discussion of limitations of earlier performance measures. Performance implications of pointers vs. arrays.

8 CS609: Algorithms for Differential Equations

Level:	MSc
Credit Rating:	15
Degrees:	ACS, Computational Science (and others if qualified)
Pre-requisites:	CS607, CS608
Pre-course work:	40 hours: 30% introductory lab, 70% preparatory reading
Taught week:	40 hours: 50% lectures, 50% supervised lab
Post-course work:	40 hours: 100% mini-project
Assessment:	35% exam, 30% lab, 35% mini-project
Lecturer(s):	Dr T L Freeman, Dr M Mihajlovic, Mr G D Riley
Limit on numbers:	50 participants

Introduction

Aims

To introduce numerical algorithms for the solution of ODEs and PDEs and to introduce the fundamental algorithmic properties of accuracy, stability and convergence.

Learning Outcomes

A student completing this course unit should:

1) have an understanding of the Numerical Analysis issues of algorithms for ODEs and PDEs (accuracy, stability, convergence); (A)
2) have an understanding of the performance implications of algorithmic developments (algorithmic efficiency). (D)

Assessment of learning outcomes

Learning outcomes (1) and (2) are assessed by examination, in the laboratory and via the mini-project.

Contribution to programme learning

A1, A2, B2, C1 and C3.

Reading list and supporting material

I.K.Eriksson, D.Estep, P.Hansbo & C.Johnson, Computational Differential Equations, Cambridge, 1996.

Special resources needed to complete the course unit

Detailed syllabus

Initial Value Ordinary Differential Equations Runge-Kutta methods and Multistep methods; accuracy, convergence and stability; error control; numerical examples.

Elliptic Partial Differential Equations Finite difference and finite element methods; solution of large systems of algebraic equations; numerical examples.

Parabolic Partial Differential Equations Finite difference methods and method of lines; analysis of error; numerical examples.

Hyperbolic Partial Difference Equations Methods based on characteristics; convection-diffusion problems; analysis of error.

9 CS611: System Construction Using B

Level:	MSc
Credit Rating:	15
Degrees:	ACS
Pre-requisites:	None
Pre-course work:	40 hours: Unsupervised reading: review of discrete mathematics
Taught week:	40 hours: Lectures and supervised labs
Post-course work:	40 hours: Unsupervised project using B
Assessment:	33% exam, 66% project
Lecturer(s):	Dr. Richard Banach
Limit on numbers:	50 participants

Introduction

It has been known for a long time that in theory, the whole activity of system specification, refinement and implementation, could be integrated into a comprehensive mathematical theory and could be supported by industrial strength system development tools. However an actual realisation of this supposition has been slow to appear. The B-Toolkit is a system that mechanises support for the B method of rigorous whole-system software construction in a fully integrated manner. The method is particularly suited to the building of the kind of state machines found at the heart of many safety critical software control systems. The course is an introduction to this methodology.

Aims

The aim of the course is to give a solid grounding in the B method using the B-Toolkit, concentrating on formal model building aspects, and giving exposure to the other elements of the theory, enabling students to go on to participate in the rigorous building of real systems.

Learning Outcomes

A student completing this course unit should:

1) have an understanding of the role of formal software development and its advantages and disadvantages (A)
2) have an understanding and knowledge of the essential elements of the B-Method (A, B and C)
3) be able to develop small applications using the B-Method (B and C)
4) be able to undertake an investigation of a larger application using the B-Method (B, C and D)
5) be able to prepare a technical report on a study performed using the B-Method (C and D)
6) be able to work effectively as a member of a group to undertake a project using the B-Method (D)

Assessment of learning outcomes

Learning outcomes (1)-(3) are assessed by the examination and the project.
Learning outcomes (4)-(6) are assessed by the project alone.

Contribution to programme learning

A2, B2, B3, C1, C3, C4, D1, D3

Reading list and supporting material

Wordsworth J.B. Software Engineering with B. Addison-Wesley, 1996. Compulsory Purchase.

Schneider S. The B-Method, an Introduction. Palgrave, 2001.

Lano K., Haughton H. Specification in B: An Introduction Using the B Toolkit. Imperial College Press, 1996.

Abrial J.R. The B Book. Cambridge University Press, 1996.

Jacky J. The Way of Z. Cambridge University Press, 1997.

B-Toolkit documentation. (Loaned to students for the duration of the course.)

Special resources needed to complete the module

The B-Toolkit software.

Detailed syllabus

Preliminary work: Review of Discrete Mathematics. Students should thoroughly familiarise themselves with Appendix A of Wordsworth's book, and with the notations (both "mathematical" and "ascii") for the concepts therein, described in Appendix B1 of the same book. Remaining time may be spent as follows. (1): Reading remaining parts of the Wordsworth book, from the beginning. (2): Familiarisation with abstract modelling via discrete mathematics. Beyond what is to be found in Wordsworth's book, the book by Jacky gives many good examples of abstract specification in Z (Z uses very similar discrete mathematics notation to B, though Z is a different system). Students may find it usefull to consult this book.

DAY 1: Getting started with the B-Toolkit. Abstract Machine Notation and Generalised Substitutions. Simple specification, animation, document generation.

DAY 2: Writing whole specifications. Structuring techniques. Development overviews.

DAY 3: Proof Obligations. The Autoprover and Interprover. Introduction to Implementations, library machines, interface generation.

DAY 4: More on Implementations and Refinements.

DAY 5: Structuring large developments. Project assignment and start of project work.

Assessed practical: Completion of project and hand in of generated documentation.

10 CS612: Automated Reasoning

Level:	MSc
Credit Rating:	15
Degrees:	ACS
Pre-requisites:	None
Pre-course work:	40 hours: preparatory reading and exercises
Taught week:	40 hours: 50% lectures, 50% supervised lab
Post-course work:	40 hours: 90% unsupervised lab or assignment
Assessment:	33% exam, 33% Taught Week Labs and Exercises, 33% Post-course work
Lecturers:	Dr. Alan Williams and Professor Peter Aczel
Limit on numbers:	50 participants

Introduction

Logic, the study of reasoning, plays an important role in theoretical computing science and many of the practical areas of computer science such as systems development, computer hardware, data bases, cognitive science, AI and logic programming. In particular an important part of the systems development process concerns reasoning about the behaviour of systems. Computer-based tools for systems development should support this reasoning; i.e. should support the construction of formal proofs.

Aims

This course unit aims to provide an introduction to classical logic and the practice of computer-assisted proof construction. There is no assumption that students are already familiar with the subject. The course aims to:

- provide students with an understanding of classical logic and underlying theory necessary for applying automated reasoning
- study a range of techniques for reasoning which are appropriate for automation
- allow students to use automated reasoning tools for constructing proofs

Learning Outcomes

A student completing this course unit should:

1. have knowledge and understanding of the syntax and semantics of classical propositional and predicate logic (A).
2. have knowledge and understanding of some of the theory underlying the B-Tool proof assistant (introduction) (A and C)
3. have knowledge and understanding of a selection of logically-based applications (A and B)
4. be able to use standard proof systems, in particular semantic tableaux, resolution, natural deduction and sequent calculus (B).
5. be able to use B-Tool when applied to simple problems (C)

Assessment of learning outcomes

- Learning outcomes 1,2,3,4 will be assessed by exam
- All learning outcomes will be assessed via exercises in the taught week

Contribution to programme learning

A1 have knowledge and understanding of the syntax and semantics of classical propositional and predicate logic (Learning Outcome 1)

A2 have knowledge and understanding of some of the theory underlying the B-Tool proof assistant (introduction) (Learning Outcome 2)

A2 have knowledge and understanding of a selection of logically-based applications (Learning Outcome 3)

B3 be able to use standard proof systems, in particular semantic tableaux, resolution, natural deduction and sequent calculus (Learning Outcome 4)

C3 be able to use the B-Tool when applied to simple problems (Learning Outcome 5)

Reading list and supporting material

There is no single book covering all material, but the following gives a good introduction to logical systems and reasoning methods:

– Kelly, J., *The Essence of Logic*, PHI

Notes will made available to cover the B-Tool tool and all of the various topics presented in the course. Together with Kelly, these give a good coverage of the material:

– Rajeev Goré and Martin Peim, *Automated Reasoning: Notes for MSc Course Unit CS612*

There are many other text books available on the topics covered. The following gives a selection of those that would be useful to refer to:

– Ben-Ari, M., *Mathematical Logic for Computer Science* Prentice Hall, 1993
– Hamilton, A. G., *Logic for Mathematicians.* Cambridge University Press, revised 1990.
– Sterling, L. and Shapiro, E., *The Art of Prolog*, MIT.

Further details will be presented in the taught week.

Detailed syllabus

The following lists topic to be covered in the pre-course work and taught week. The numbers of lectures for each topic are given in brackets. If the topic is to be covered in the pre-course work or in the supervised labs, then this is also indicated:

Introduction and motivation: [**Pre-Course, 1**] including example problems, problem representation via logic, computer assisted reasoning in mathematics.

Propositional logic: [**Pre-Course**]

theory, language, models, validity and satisfiability
proof/inference rules, soundness and completeness,
reasoning methods: truth tables, proof by contradiction, natural deduction.

Propositional Logic Reasoning using Resolution: [**1**] normal forms, resolution

First-order Predicate Logic Introduction: [**1**] quantifiers, first order models, validity and satisfiability

First-order Reasoning using Resolution [2]

normal forms,
skolemization: elimination of quantifiers
unification,
resolution
strategies for improving efficiency

Logic Programming: [2, supervised labs] Horn clauses, SLD resolution, Prolog

Natural deduction Theory: [Pre-Course, 2]

propositional and predicate rules and their formulation
heuristics
relationship with other systems
sequent calculus

The B-Tool Tool: [2, supervised labs]

Representation in B-Tool of the natural deduction rules for propositional and predicate logic and natural deduction proofs, using those rules, in forwards reasoning style and also in backwards reasoning style via the B-Tool tactics. Introduction to some of the theory behind B-Tool.

11 CS616: Knowledge Representation and Reasoning

Level:	MSc
Credit Rating:	15
Degrees:	ACS/CS for 2nd semester course units
Pre-requisites:	Some knowledge of logic and formal methods.
Pre-course work:	40 hours: preparatory reading and exercises.
Taught week:	40 hours: 60% lectures, 40% supervised lab.
Post-course work:	40 hours: 90% unsupervised lab or assignment
Assessment:	40% exam, 30% Taught Week Labs and Exercises, 30% Post-course work.
Lecturers:	Dr. Ulrike Sattler and Dr. Renate Schmidt
Course unit webpage:	http://www.cs.man.ac.uk/~schmidt/CS616/
Limit on numbers:	50 participants

Introduction

For many applications, specific domain knowledge is required. Instead of coding such knowledge into a system in a way that it can never be changed (hidden in the overall implementation), more flexible ways of representing knowledge and reasoning about it have been developed in the last 10 years. These approaches are based on various extensions of classical logic: modal logic, agents logics, or description logics. They can be used to reason about the terminology of a domain or the behaviour of systems. Computer-based tools can then use this kind of reasoning to support the user. In particular description logics have recently been used as foundational tools for the semantic web.

Aims

This course unit aims to provide an introduction to various extensions of classical logic, how to formalise knowledge and questions about this knowledge in these logics and how to use automated reasoning systems for answering these questions. Students should have some knowledge about logic and will deepen it in the first pre-course week. The course unit aims to:

- provide students with an understanding of different kinds of knowledge and the logics developed to represent this kind of knowledge together with the underlying theory necessary for applying automated reasoning systems (based on propositional, first order, modal, and description logic)
- study a range of techniques to formalise and represent knowledge within these logics, and, finally,
- allow students to use various automated reasoning tools to reason about knowledge represented in these logics.

Learning Outcomes

A student completing this course unit should:

1) have knowledge and understanding of the syntax and semantics of modal, description, and temporalised description logics, defaults, and formal concept analysis (A)
2) be able to formalise and represent knowledge in these logics and relate questions concerning this knowledge to logical reasoning problems (A and B)
3) have knowledge and understanding of a selection of logic-based applications (A and B)
4) be able to use standard proof systems, in particular Hilbert-style deduction and a translation-based approach for modal logics, subsumption algorithms for description logics, and the attribute exploration algorithm (B)
5) be able to use various systems (SPASS, ICOM) and apply them to solve problems (C)

Assessment of learning outcomes

Learning outcomes (1), (2), (3), (4) will be assessed by exam. All learning outcomes will be assessed via exercises in the taught week and the post-course work.

Contribution to programme learning

A1, A2, B2, B3, and C3.

Reading list and supporting material

There is no single book covering all the material, but the following gives a good introduction to logical systems and reasoning methods: Kelly, J., *The Essence of Logic*, PHI.

Notes will made available to cover the systems (SPASS and ICOM) and all of the various topics presented in the course.

There are many other textbooks available on the topics covered. The following gives a selection of those that would be useful to refer to:

Modal Logic:

Recommended reading is Chapter 3 on modal logic and its applications in the book by M. Huth and M. Ryan, *Logic in Computer Science: Modelling and Reasoning about Systems*, Cambridge University Press, 2000.

Description logics:

Recommended Reading is the chapter by F. Baader and W. Nutt, *Basic Description Logics*, in the Description Logic Handbook (edited by F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider, Cambridge University Press, 2002, pp. 47–100).

Further details will be presented in the taught week.

Detailed syllabus

The following lists topics to be covered in the pre-course work and taught week. The number of lectures for each topic are given in brackets. If the topic is to be covered in the pre-course work or in the supervised labs, then this is also indicated:

Introduction and motivation [Pre-Course]. Including example problems, problem representation via logic, computer assisted reasoning in mathematics.

Elementary set theory [Pre-Course]. What is a set, a relation, a function, set operations (intersection, union, etc), properties of binary relations (reflexivity, symmetry, transitivity, etc).

Propositional logic [Pre-Course]. Theory, language, models, validity and satisfiability, inference rules, soundness and completeness, reasoning methods: truth tables, proof by contradiction.

First-order logic [Pre-Course]. First order logic formulae, their meaning, validity and satisfiability, translating between natural language and first-order logic.

Early knowledge representation formalisms [1]. Nonmonotonic inheritance networks, frame-based systems.

First-Order Logic [2, supervised lab]. First order logic formulae, their meaning, reasoning problems, useful normal forms, inference calculus, undecidability and semi-decidability.

Modal Logic: Representation and reasoning on the semantical level [4.5, supervised lab]. Modal logic, possible worlds semantics, model checking, satisfiability and validity, correspondence theory.

Modal Logic: Reasoning calculi, agent applications [4, supervised labs]. Logically omniscience problem, belief logic, epistemic logic, deduction in Hilbert systems, deduction via translation to first-order logic.

Description logics [3.5, supervised labs]. Language of description logics, meaning of description logic statements, reasoning calculi, introduction to the semantic web and ontologies using description logics.

Icom [2, supervised labs]. EER diagrams, relationship between EER diagrams and description logic, reasoning about EER diagrams.

Non-standard reasoning services in description logics [2, supervised labs]. Least common subsumers, most specific concepts, and their usage in description logic applications.

Temporal logic [3, supervised labs]. The temporal logic LTL, its extension to temporalised modal and description logics, their applications.

Defaults, in propositional and first order logic [3, supervised labs]. Defaults, motivation for ordered defaults, e.g. in description logics, their applications.

Coursework

Exercises and assignments are of varying difficulty – those in the teaching week are aimed to consolidate the material of the lectures and are thus easier. Some exercises and assignments are to be done with pencil and paper, some will require the use of tools (SPASS for the ML, ICOM for the DL part).

For the post-course work you will be given a selection of topics from which you choose one. This work may involve writing a program, formalising problems, using reasoning tools for solving such problems, a case study on some research in one of the areas, or a mixture of these.

Additional Information

Additional information may be found at the course unit webpage.

12 CS617: Interactive System Design Methods

Level:	Advanced MSc
Credit Rating:	15
Degrees:	ACS
Pre-requisites:	Programme prerequisites only
Pre-course work:	60 hours: See below
Taught week:	Lectures, discussions, exercises, and workshops. This is usually longer Mon-Fri, and includes part of the weekends. Make sure you are available for the full period of the interactive group teaching.
Post-course work:	None (see below)
Assessment:	100% coursework
Lecturer:	Dr Mark van Harmelen, mailto: markvanharmelen@yahoo.com

Introduction

Attend this module if you want to learn about

1) modern approaches to the design of the scope, content, functionality and user interfaces to interactive systems, and
2) the theory of interactive system design methods, particularly those in the field of object-oriented human-computer interaction (oohci) design.

The module is devoted to the design of interactive systems which are modelled with the Unified Modeling Language (UML), but where the design is informed by human-computer interaction (HCI) design methods. The module is concerned with software design methods for the early stages of the system development life cycle; when the overall system scope, contents, and functionality are designed. Conventional methods neglect system usability; the oohci methods advocated in this course attend to the usability of systems by, typically, by both users and technical design staff in a participatory design team that designs a system and its interface according to task-based requirements. The team uses informal task and object modelling to record the developing design. The informal notation, understandable by all the designers, can be translated into a more structured form like UML. A significant part of the course is devoted to gaining practical experience in using participatory design methods. Other parts of the course are concerned with understanding the spectrum of oohci methods available and to improve and customise methods for particular circumstances by selection of different design techniques

This course is based on recent methods research in integrating object modelling methods with HCI design methods.

The course runs eight days, Sunday through to the subsequent Sunday. By the time the course is finished all work for the course will have been performed, having been divided into some 60 hours pre-course work, and 60 hours course work and continuous assessment.

The method of instruction is one that combines background reading, discussion, and hands-on practical application of design methods. The latter provides experiential learning in the application of oohci design methods.

The practical section of the course consists of an interactive system design problem that lasts for the duration of the course. The solution of the design example is performed by students in class using a method that combines participatory design by stakeholders together with task and object-oriented description techniques to record the developing design. By the end of the course students will complete an analysis-level object-oriented description of an interactive system and have developed a user interface for the system in a single integrated design process.

Aims

The module aims to introduce students to effective methods for interactive system design.

Learning Outcomes

A student completing this course unit should:

1) Have knowledge and understanding of the principles of interactive system design, particularly those within the oochi design tradition. (A)
2) Understand how to perform, and have experience in performing, interactive system design using a participatory oohci design method. (B2,B3,C1,C3,C4,D)
3) Have at least a reading knowledge of the Unified Modeling Language, if not the ability to write analysis level system descriptions in UML. (A2)
4) Understand what constitutes an oohci method, and have the ability to design their own oohci method. (A,B2)

Assessment of learning outcomes

Learning outcomes are assessed using two in-course techniques: (a) group exercises, (b) continuous-assessment tests.

Contribution to programme learning

This course unit contributes to the following programme outcomes: A1, A2, B2, B3, C1, C3, C4, D1, D4, and D5.

Reading list and supporting material

Reading material for the course is provided as a pre-printed pack.

UML Slides

van Harmelen, M. Object Modelling with UML for OOHCI use. Slides. 2002.

There is sufficient UML for you to be able to produce analysis level designs. If you don't know it, teach yourself UML from these slides.

HCI design slides

van Harmelen, M. Designing Graphical User Interfaces. Slides.

These slides provide enough knowledge for you to do traditional user interface design. The slides are from an old commercial course, but are relevant as introductory material which will be assumed as known in the course. Don't worry too much about the Windows specific detail where it occurs, except as a common illustration.

There is sufficient UML for you to be able to produce analysis level designs. If you don't know it, teach yourself UML from these slides.

Basics of HCI methods

Start by reading van Harmelen [2001, 10.2 and its subsections].

Invent some kind of chart or list where you can cross reference ideas which crop up in all readings in this subsection.

Norman, D.A. Cognitive Engineering. Norman DA and Draper SW (Eds). *User Centered System Design*, 1986.

This introduces you to the idea of Cognitive Engineering, where HCI designers use cognitive models as an aid to system design. The Chapter includes some ideas from cognitive psychology as to how users interact with computers. At the end of the paper Norman introduces User Centered Design (UCD). Cognitive Engineering and UCD are central to the course and underpin all hci and oohci methods, including that of the Bridge, used in the course. The book from which the chapter is drawn was broadly influential in the HCI field.

Karat, J. Evolving the Scope of User-Centered Design. Communications of the ACM. 40(1). July 1997.

An update on the loose HCI definition of UCD, building on Norman [2001]. Note how Karat defines UCD as an engineering technique to counter arguments that user need should not totally determine system design. This is a readable article that should be fully digested.

Landauer, T.K. User Centered Design Methods. Chapter from Landauer, TK, The Trouble With Computers, 1995.

This chapter plunges you into a discussion of UCD methods and their components. Skip what you find difficult. GOMS is not worth considering, as Bannon [1991, p37] notes "The practical utility of such low-level calculation models in actual design has been the subject of some debate."

Bannon, L. From Human Factors to Human Actors: The Role of Psychology and Human-Computer Interaction Studies in System Design. Chapter from Greenbaum, J., and Kyng, M. (Eds) Design at Work: Cooperative Design of Computer Systems, 1991.

Another chapter that plunges you into issues surrounding design. Again a reading that plunges you into different topics pertaining to interactive design; skip what you find difficult, but those comfortable with the material will gain from it. Two important themes are emphasised here, users as actors (somewhat different, but related, to a UML 'actor'), and participatory design. You should definitely read about these two.

Design techniques and methods

A method (often inaccurately called a 'methodology') describes how to design something. It consists of design techniques (e.g. task analysis, object modelling) and notations for recording and communicating a design. A method to one person (e.g task modelling to a task analyst) may actually be a design technique in a larger method. Methodology is what we are do to some extent in this course, the examination, evaluation and improvement of methods, and invention of new methods. The readings are diverse and try to cover just a few techniques and methods.

van Harmelen [2001] provides broad discussions of HCI and oohci design techniques and methods. When you read it list the design techniques that you encounter.

Kensing, F. and Madsen, K.H. Generating Visions: Future Workshops and Metaphorical Design, chapter from Greenbaum, J, and Kyng, M, (Eds), Design at Work: Cooperative Design of Computer Systems, 1991.

Included for you to browse (rather than read in depth) and start to perceive just how diverse design techniques are. This is a brain storming type of technique. Try to apply it to the design of some system that you have encountered.

Monk, A. and Howard, S. The Rich Picture: A Tool for Reasoning About Work Context. Interactions. March April 1998.

Muller, M.J. PICTIVE - An exploration in Participatory Design, Proc CHI 92, ACM, 1992. You will have read about participatory design by now in Bannon [1991]; PICTIVE is an early participatory design technique. The eading describes the participatory construction of prototypes using paper and other low-tech materials. Don't imagine that using the video recording technique is necessarily good, accessing video is notoriously time consuming. Skip any difficult bits at the end of the paper.

Muller, M.J. Retrospective on a year of Participatory Design using the PICTIVE Technique, Proc CHI93, ACM, 1993.

Experience of using PICTIVE, including feedback about what is good and bad.

Rudd, J., Stern, K. and Isensee, S., Low vs. High-Fidelity Prototyping Debate, Interactions, ACM, Jan 1996.

Its hard to put any particular prototyping (a design technique) paper in the reading, there are many such papers. This one contrasts and compares two different styles of prototyping.

Dayton, T., McFarland, A, Kramer, J. Bridging User Needs to OO Gui Prototype via Task Object Design, in Wood, L (Ed)

A participatory oohci method is described here. This paper is long, but you must make time to read it, because you will be using a modified form of the Bridge in class. Note in reading this that a 'task object' simply means an 'object' that the user uses in the execution of a task. Note that by concentrating first on tasks and task flows the functionality and scope of the system are defined. By then extracting objects from the task flows the contents of the interactive system can be determined. Finally a user interface is designed for the system from the task flows and objects identified earlier. Iterative design, formative evaluation of paper prototypes in different kinds of usability test are important component of this method and results in a validated design. See van Harmelen [2001] for a discussion of these techniques. Thee is a set of pdf slides produced by a previous student at www.oohci.org accessible via the Bridge page, these may help structure your reading of this paper.

Performing the bridge provides a work context for participatory designers in design sessions. As part of your pre-course work, you are asked below to try to provide four rich pictures describing the Bridge.

Evaluation techniques

The evaluation technique used in this course is formative evaluation of paper proptypes. The readings here give some idea of different techniques.

Newman, WM, and Lamming, MG. 8.5 Analysis by Cognitive Walkthrough. 8.6 Heuristic Evaluation. 14.5 Walkthrough Analysis, Design Problems. in Newman, WM, and Lamming, MG, Interactive System Design, 1995.

Some evaluation techniques.

Wright, P., and Monk, A. A cost-effective evaluation method for use by designers. Int. J. Man-Machine Studies, 35(6), 1991, 891-912.

Keen students find and read this.

Object modelling and human-computer interaction (oohci) design

van Harmelen, M., Designing with oo&hci Methods. Summary chapter from van Harmelen, M, (Ed), Object Modeling and User Interface Design: Designing Interactive Systems. 2001.

Back to design philosophies (UCD, Cognitive Engineering), design techniques and methods. This time integrating the fields of HCI and object modelling in oohci methods, the subject of this course. Oohci methods are called oo&hci methods in this chapter. If you find the UML hard at the end of the chapter but have read the rest of it don't worry too much. However you will gain most by reading the class diagrams.

Participatory practice

Muller, MJ, Haslwanter, JH, and Dayton, T. Participatory Practices in the Software Lifecycle. In Hellander MG, Landauer, TK, and Prabhu, PV, (Eds), Handbook of Human-Computer Interaction, 2nd Ed, 1997.

Read 11.1 through 11.5. This picks up and discusses participatory themes that run through much of the reading.

Special resources needed to complete the module

None.

Detailed syllabus

Introductory topics

Overview of the interactive system development lifecycle and its requirements. Methods and methodology.

Essential UML notation for high-level system specification purposes, including object, class, use case, sequence, and collaboration diagrams.

Responses to the development lifecycle by the software engineering and human-computer engineering communities. Why traditional object-oriented design methods fail to adequately address interactive system design.

Human-computer interaction and interaction design.

Basics of object-oriented human computer interaction (oohci) design methods.

Participatory design

Some Participatory Design (PD) methods. Factors contributing to successful PD. Facilitation, Brainstorming, Rich Pictures.

An example oohci method

The Bridge; what each stage of the method does. Use of a modified form of the bridge.

A formal treatment of oohci methods

The final day of the course is devoted to: Assessing the practical techniques used and discussing further method development. Necessary and optional components of oohci methods. Situating the Bridge in a UML description of oohci methods. Process issues. Designing oohci methods to suit project or organisational needs.

Pre-course work

Read the pre-course reading as printed and distributed by the School (collect from room 2.10).

Familiarise yourself with web-based resources: Examine www.oohci.org and www.primaryview.org. Search more broadly for information on participatory design, facilitation and related matters as apparent from the pre-course reading and your own web-based research. Keep a record of interesting information that you find, this is potentially a contributor to your course marks.

This is a time breakdown of what you should do during 60 hours pre-course work:

20 hrs: Read, practice and learn object modelling using UML notes

28 hrs: Read supplied reading pack (see above).

4 hrs: Draw a rich picture that describes use of The Bridge. Then draw three rich pictures, one for each stage of The Bridge. Rich Pictures are described in the reading pack.

8 hrs: Spread over the pre-course work time period: research the web, noting interesting resources for cs617.

The last two items should be handed in at the very start of the course.

Pre-course reading is the topic of the first continuous assessment test.

Post-course work

Thanks to the longer and more intensive nature of the course, there is no post-course commitment, i.e., there is neither a post-course project nor an exam.

13 CS618: Object-Oriented Analysis and Design (for CS students)

Level:	MSc
Credit Rating:	15
Degrees:	CS (exceptionally ACS/CompSci)
Pre-requisites:	Knowledge and/or experience of programming in at least one high-level imperative language (object-oriented or structured eg Java, Smalltalk, Eiffel, C, Ada, Modula or C++)
Pre-course work:	40 hours
Taught week:	40 hours (50% lectures, 50% laboratory exercises in groups)
Post-course work:	40 hours (completing and preparing documentation for laboratory solutions)
Assessment:	By on-line examination
Lecturer:	Mr Mike O'Docherty (mod@cs.man.ac.uk)

Aims

This course unit aims to describe what object-oriented (OO) software is all about. More specifically, to teach the concepts, tasks and notation (using UML).

In labs, you will try each of the tasks, working in teams of three or four. One or more members of each team will be expected to present results to the class at regular intervals.

There will be no attempt to cover the specifics of implementation in any particular programming language. Instead, we will assume that a pure OO language is available to implementors (e.g. Java, Smalltalk, Eiffel).

The material in this course will make sense to any programmer, i.e someone who knows what a computer is and has written programs in at least one of the languages listed above. Other than that, you will need to possess certain human qualities, such as the ability to listen, think, discuss and experiment.

Learning Outcomes

After successful completion of the course unit, a student will (1) understand how to design software in an object-oriented manner (A and B), (2) have mastered UML as a notation to support this design (B and C), (3) have undertaken a reasonably sized OO design in UML as part of a team-work exercise (B and D).

Assessment of learning outcomes

During the week, each group will be expected to prduce a draft "Project Workbook" containing documents produced during the development of a solution to the labs. The end product of each group will be wotrth up to 30be expected to produce a final version of their workbook for assessment - this will be worth up to 30

The exam comprises a 90-minute multiple-choice test, taken on-line. The idea behind the test is to check that you have learnt enough without the inefficiency and subjective marking inherent in a written examination.

Contribution to programme learning

A1, A2, B2, C1, C3, D1.

Reading list and supporting material

Everything you need to know in order to succeed on the course unit will be presented to you during the course, or it will be practised by you in the laboratorie exercises. Therefore, no prior or background reading should be necessary (unless you feel that you need extra work to meet the pre-requisites).

Books may be recommended nearer the time or during the course unit itself, but reading them will be optional.

Special resources needed to complete the course unit

None.

Detailed syllabus

The course covers life-cycle and tasks for OO software development, up to, but not including, the actual writing of code:

- Object Concepts
  - Objects
  - Classes
  - Inheritance
  - ObjectitemOriented Type Systems
- Software Development Mehodology
  - Engineering or invention?
  - Example Artifacts using UML
  - CRC Cards
- Requirements Capture
  - Introduction
  - Business Perspective
  - Developer Perspective
- Analysis
  - Introduction
  - Static Analysis
  - Dynamic Analysis
- System Design
  - Introduction
  - Networked System Topologies
  - Choosing Technologies
  - Partitioning Software
- Subsystem Design
  - Designing the Business Logic
  - Persistence using a Relational Database
  - Finalizing the User Interfaces
  - Designing the Business Services
  - Thread Safety

- Code Specification
  - Background
  - Object-Oriented Specification
  - Design by Contract
  - Informal Specification in Java

UML Notation will be used throughout the course unit.

14 CS619: Extreme Java

Level:	MSc
Credit Rating:	15
Degrees:	ACS/CompSci/CS
Pre-requisites:	See below
Pre-course work:	40 hours (ensuring that you meet the pre-requisites)
Taught week:	40 hours: 50% lectures, 50% supervised lab
Post-course work:	40 hours (finishing the labs/revising for the test)
Assessment:	90 minute on-line test (under examination conditions)
Lecturer:	Mr Mike O'Docherty. Contact E-Mail: mod@cs.man.ac.uk

Aims

This course unit aims to identify all the high-level facilities in the Java 2 SDK, Standard Edition, and to show how these are being used to re-engineer the software industry.

The course unit also aims to encourage a deeper understanding of life as a senior Java programmer (by getting you to work with realistic code rather than noddy examples).

As "Extreme" in the title suggests, the content is non-trivial. Don't be put off, however: we are just trying to make sure that you don't arrive on the course without meeting the pre-requisites!

Learning Outcomes

After successful completion of the course unit, students will have mastered the appropriate tools for the design and implementation of a medium to large scale enterprise application, using the full power of Java's portability and network awareness. (A, B and C)

Assessment of learning outcomes

The course unit is assessed by a 90 minute on-line test under examination conditions. The test will be a thorough-going examination of the material and skills taught on the course unit (learning outcomes A1, A2, B2, B3, C3).

Contribution to programme learning

A1, A2, B2, B3, C3.

Reading list and supporting material

Before attending this course, make sure that you understand Java and it's related technologies to the level described in books (1) and (2) below, (more detail is given in the Pre-requisites section below).

Before you buy any book, remember that most things you might want to learn are available on the Internet for nothing, you just might have to look a little harder.

In the list below, (1) is a language/libraries book (other introductory texts from Addison-Wesley/Prentice-Hall/O'Reilly would proably be just as good), (2) is a related-technologies book (now a little old, but it sets the seen well), (3) is a free language/libraries tutorial (also available as a book).

1. Peter van der Linden, Just Java And Beyond 1.1 (3rd Edition), Prentice Hall, 1997
2. Peter van der Linden, Not Just Java, Prentice-Hall, 1997
3. http://java.sun.com /tutorial

Special resources needed to complete the course unit

No special resources are required.

Detailed syllabus

1. Performance and Programming Style
   - What is Performance?
   - Improving Performance
   - Programming Tips and Tricks
2. Persistence in Java
   - Serialization
   - Java Database Connectivity (JDBC)
3. Common Object Request Broker Architecture (CORBA)
   - CORBA and IIOP
   - Hello World
   - Interface Definition Language (IDL)
   - Mapping to Java
   - Host Implementation
4. Reflection and JavaBeans
   - Inspecting Classes at Run Time
   - Working with Inspected Classes
   - What is a Software Component?
   - Implementing Software Components
5. Java Foundation Classes (JFC)
   - Swing Components
   - Swing Features
   - Accessibility
   - JFC and the Abstract Windowing Toolkit (AWT)
6. Remote Method Invocation (RMI)
   - RMI Architecture
   - RMI API
7. Java Native Interface (JNI)
   - Calling Native Functions from Java
   - Calling Java Methods from Native Code
   - Manipulating Java objects in Native Code
   - Garbage Collection and JNI
   - Invoking a Virtual Machine from Native code
8. Security
   - Security Overview
   - jar - Managing Java Archives
   - keytool - Managing Keys and Certificates
   - jarsigner - Signing and Verifying JAR Files
   - policytool - Managing Security Policies
9. Overview of Optional Packages
   - The Optional Packages
   - Other Players

Pre-requisites

Good knowledge and/or experience of the Java language and the basic packages (java.lang, java.util, java.text, java.awt, java.applet, java.io, java.net, java.math) is necessary; as is a thorough understanding of OO concepts and programming; and a working knowledge of OO design and analysis.

A good understanding of the technologies surrounding Java. e.g. Internet, relational databases is also required.

For those of you who don't meet the pre-requisites, an alternative to the reading material listed above is to attend one of John Sargeant's introductory courses. (Details elsewhere in this syllabus.)

15 CS620: Electronic Instrumentation Systems MSc: Design Study and Project (EIS)

Syllabus under preparation: see the EIS Programme Handbook[3] (in MSWord) for details.

[3] http://www.cs.man.ac.uk/Postgrad/ACS-CS/webpages/handbook/eis/eis.doc

16 CS621: Instrumentation Electronics (EIS)

Syllabus under preparation: see the EIS Programme Handbook[4] (in MSWord) for details.

[4] http://www.cs.man.ac.uk/Postgrad/ACS-CS/webpages/handbook/eis/eis.doc

17 CS624: Mobile Computing

Level:	MSc
Credit Rating:	15
Degrees:	ACS/CS
Pre-requisites:	Basic mathematics
Pre-course work:	40 hours
Taught week:	40 hours: lectures and laboratories
Post-course work:	40 hours assessed practical
Assessment:	67 % laboratory and 33 % exam
Lecturer(s):	Prof. Barton, Dr. Zhang, Mrs. Costen
Limit on numbers:	50 participants

Aims

To impart an understanding of fundamental concepts underlying current developments in mobile communication systems and wireless computer networks.

Learning Outcomes

At the end of the course, students will have acquired the following knowledge and skills.

1) Understanding of characteristics of radio propagation and interference in multipath propagation and channel model description (A1,A2)
2) Understanding of a range of digital transmission systems as used for applications in mobile telephony and wireless computer networks, pulse shaping and equalisation techniques (A1,A2)
3) Understanding of the issues and techniques used in the design of Medium Access Control protocols for wireless Networks (A1,A2)
4) Understanding of the systems, protocols and mechanisms to support mobility for mobile internet users (A1,A2)
5) The ability to investigate fundamental aspects of transmission and modulation by writing MATLAB programs. The experience of using an industrial standard network simulation package, such as OPNET(B1,C1,C2,D4)

Assessment of learning outcomes

The first four outcomes are assessed through examination; all the outcomes are assessed through an assessed practical project.

Contribution to programme learning

A1,A2, B1,C1,C2,D4.

Reading list

- J.Schiller, Mobile communications, ISBN: 0-321-12381-6, Addison-Wesley, 2003

Supplemental books

- T.S. Rappaport, Wireless communications; Principle and Practice, ISBN: 0-13-375536-3
- A S. Tanenbaum, Computer Networks (Fourth Edition), Publisher: Prentice Hall PTR; ISBN: 0130661023; August, 2002.

Detailed Syllabus

Introduction to wireless networking. Advantages and disadvantages of wireless networking

Characteristics of radio propagation. Fading, Multipath propagation

Introduction to digital transmission. Definition of bit-rate and signalling rate. Introduction to synchronous transmission. The need for pulse shaping, synchronisation and line-coding. Calculation of bit-error probabilities when the channel is affected by the addition of Gaussian noise.

Narrowband digital modulation. The need for modulation. Binary and multi-level (M-ary) amplitude-shift keying (ASK), frequency-shift keying (FSK) and phase-shift keying (PSK).

Wideband modulation techniques to cope with intersymbol interference Direct sequence spread spectrum Adaptive Equalization Orthogonal frequency division multiplex

Medium Access Control (MAC). MAC protocols for digital cellular systems such as GSM. MAC protocols for wireless LANs such as IEEE802.11 and HIPERLAN I and II. The near far effect. Hidden and exposed terminals. Collision Avoidance (RTS-CTS) protocols.

Protocols supporting mobility. Mobile network layer protocols such as mobile-IP, Dynamic Host Configuration Protocol (DHCP). Mobile transport layer protocols such as mobile-TCP, indirect-TCP. Wireless Application Protocol (WAP).

18 CS625: Information Storage Systems (EIS)

Syllabus under preparation: see the EIS Programme Handbook[5] (in MSWord) for details.

[5] http://www.cs.man.ac.uk/Postgrad/ACS-CS/webpages/handbook/eis/eis.doc

19 CS626: Sensors and Systems (EIS)

Syllabus under preparation: see the EIS Programme Handbook[6] (in MSWord) for details.

[6] http://www.cs.man.ac.uk/Postgrad/ACS-CS/webpages/handbook/eis/eis.doc

20 CS627: The Measurement Environment (EIS)

Syllabus under preparation: see the EIS Programme Handbook[7] (in MSWord) for details.

[7] http://www.cs.man.ac.uk/Postgrad/ACS-CS/webpages/handbook/eis/eis.doc

21 CS628: Signal Processing (EIS)

Syllabus under preparation: see the EIS Programme Handbook[8] (in MSWord) for details.

[8] http://www.cs.man.ac.uk/Postgrad/ACS-CS/webpages/handbook/eis/eis.doc

22 CS629: Electronic Instrumentation Systems: Optional Subjects (EIS)

Syllabus under preparation: see the EIS Programme Handbook[9] (in MSWord) for details.

[9] http://www.cs.man.ac.uk/Postgrad/ACS-CS/webpages/handbook/eis/eis.doc

23 CS631: Computational Biology - The application of computer science to the problems of post-genome biology

Level:	MSc
Credit Rating:	15
Degrees:	Advanced and CS
Pre-requisites:	A knowledge of modern biology is not a course prerequisite.
Pre-course work:	40 hours: preparatory reading
Taught week:	40 hours: 18 lectures, 4 hours tutorials, 18 hours supervised problem solving sessions.
Post course work:	40 hours. Unsupervised problem solving and project work.
Assessment:	100% coursework.
Lecturer(s):	Prof. A. Brass.
Limit on numbers:	50 participants

Introduction

Biology is currently undergoing a revolution. The success of the human genome project and other high-throughput technologies is creating a flood of new data. Capturing, interpreting and analysing this data provides real and significant challenges for computer scientists. This course will use biology as an exciting application domain for a wide range of CS techniques that have been developed on the course.

The course is organised in 4 sections:

1. basic introduction to modern biology and bioinformatics
2. data capture
3. data delivery
4. data analysis

Each section will commence with a short taught component delivered as research seminars. Assessments will be based on a short written report and presentations based on a case study that will be introduced at the start of the course.

Learning outcomes

A student successfully completing this unit will have:

1) A basic understanding of the computational needs of modern biology
2) Developed an understanding of the problems inherent in communicating with scientists from a different discipline
3) Developed the ability to reflect upon and synthesize a range of computational techniques to develop effective problem solving strategies in an unfamiliar problem domain.
4) Developed the ability to communicate these strategies to non-specialists

Assessment

Learning outcomes will be assessed in the reports and presentations based on the case-study.

Detailed Syllabus

- Intro to Biology
- Intro to Biology - the central dogma (2 hours)
- Intro to genomics (2 hours)
- Biology databases (2 hours)
- Data capture

- capturing microarray data (1 hour)
- proteomics seminar (1 hour)
- the gene ontology (1 hour)
- resource meta-data (1 hour)
- Data delivery
- HCI and bioinformatics (2 hours)
- Dealing with heterogeneous, distributed data. (2 hours)
- bioinformatics and the grid (2 hours)
- Data analysis
- Integrated approaches to post-genome data (2 hours)

24 CS632: Computer Animation

Level:	MSc
Credit Rating:	15
Degrees:	Advanced Courses (Others by arrangement with Course Director)
Pre-requisites:	Students taking this course must have some previous experience of computer graphics programming.
Pre-course work:	40 hours: 25% introductory lab, 75% preparatory reading
Taught week:	40 hours: 50% lectures, 50% supervised lab
Post-course work:	40 hours
Assessment:	50% exam, 50% practical work
Lecturer(s):	George Leaver (MC) with Nigel John (University of Wales, Bangor)
Limit on numbers:	25 participants

Aims

This course unit will provide a detailed introduction to the common algorithms and techniques used to create 3D computer animation. The principles of traditional animation will be briefly covered, and the techniques applicable to the computer medium will be highlighted. Also covered will be interpolation techniques, natural phenomena, modeling, and animation of artic ulated figures. Case Studies from leading computer animation studios will be presented, such as Pixar (Toy Story, Luxo Jr, Geri's Game, ...), Square Studios (Final Fantasy), and Blue Sky Studios (Bunny, Ice Age).

Students taking this course unit will be expected to have some prior familiarity with computer graphics concepts and programming. The course unit is aimed at computer scientists interested in the technical aspects of computer animation and no artistic skills are required. Use of tools such as 3D Studio Max and Soft Image are outside the scope of this course unit.

The number of students taking this course unit will be limited to 25.

Learning Outcomes

A student successfully completing this course unit will:

1. Have a knowledge and understanding of leading-edge computer graphics as applied to the computer animation medium.
2. Have a knowledge and understanding of the technology behind the latest generation of computer animation films.
3. Be able to implement standard computer animation programming technique s.

Assessment of learning outcomes

Learning outcomes (1), and (2) are assessed by examination, learning outcomes (3) in the laboratory.

Contribution to programme learning: A1, A2, B1, B3, C1, D1.

Reading list and supporting material

Rick Parent, "Computer Animation Algorithms and Techniques", Morgan Kaufmann Publishers, ISBN 1-55860-579-7;

John Vince, Essential Computer Animation Fast;

The RenderMan Companion: A Programmer's Guide to Realistic Computer Graphics, Steve Upstill, Addison-Wesley, 1990, ISBN 0-201-50868-0.

John Lasseter, Tricks to Animating Characters with a Computer (will be supplied);

G. Scott Owen's Renderman Tutorial for Blue Moon Rendering Tools (will be supplied).

Special resources needed to complete the course unit

The laboratory exercises will use BMRT (public domain Renderman compiler), and require PC with graphics cards.

Detailed syllabus

Lectures:

- Introduction to the course unit
- Overview of Computer Animation
  - Principles of Traditional Animation
  - History and Classification
  - Examples
- Computer Graphics Primer
  - Rendering Techniques
  - Renderman
- Interpolation Techniques
  - Function Curves
  - Motion Paths
- Animation of Articulated Objects
  - Forward Kinetics
  - Inverse Kinetics
- Advanced Techniques
  - Particle Systems
  - Soft objects
  - Natural phenomena
  - Automation (dynamics, motion capture)
- Case Studies
  - Pixar
  - Final Fantasy
  - Other key examples from recent productions.

Laboratory Sessions:

- Moving rigid body using function curves.
- Implementing Inverse kinematics
- Facial Animation

25 CS633: Computing Shape

Level:	MSc
Credit Rating:	15
Degrees:	ACS/CompSci
Pre-requisites:	None
Pre-course work:	40 hours
Taught week:	40 hours
Post-course work:	40 hours
Assessment:	33% exam, 67% practical exercise
Lecturer:	Terry Hewitt and Lin Fenqiang
Limit on numbers:	50 participants

Aims

For many years, computers have been used to store and manipulate the geometry of curves and surfaces. The aim of this course unit is to see how mathematics is applied to develop a powerful and effective representation scheme based upon non-uniform rational B-Splines (NURBS). We then go on to the fundamental problem: given a representation of, for example, a curve, produce a series of straight line segments that can fool an observer into thinking that he or she can see a smooth curve. This list of points may appear as pixels or lines on a computer display, or may be used to make, for example, car parts, helicopter rotor blades, as well as being critical to the performance of virtual reality systems. This is followed by discussion of a number of techniques to help visualize smoothness, and the course unit finishes with a lecture and demonstration of the current research work in this area. The aim is to provide an understanding of representing curves and surfaces within computers, and the critical issues in using those representations to provide numerically stable and accurate algorithms

Learning Outcomes

A student completing this course unit should:

Understand the mathematical basis for the representation of curves and surfaces in computation, (A) understand and use approximations to curves, (A and B); understand and use relevant algorithms, (A and B); be able to apply a range of techniques to a variety of applications in computer graphics, (A, B and C).

Assessment of learning outcomes

The practical skills of using representations of curves and surfaces are assessed in the practical exercises. The examination tests the understanding and knowledge described above.

Contribution to programme learning

A1, A2, B2 and C1.

Reading list and supporting material

- Ding, Quilin and Davies, Surface engineering geometry for computer aided design and manufacture. Ellis-Horwood, 1987.
- Farin, G. Curves and surfaces for computer aided geometric design. 3rd edition. Academic Press, 1993.

Special resources needed to complete the course unit

No special resources are required for this course unit.

Detailed syllabus

- Introduction, basic theory of curves surfaces: Different mathematical representation schemes (non-parametric explicit and implicit and parametric) their advantages and disadvantages. Drawing an ellipse (discussion). Simple tensor products surfaces. Linear interpolation and piecewise linear interpolation. Lecture hours (2)
- Transformations: Homogeneous coordinates; matrix representation of 2D and 3D linear and affine transformations and perspective; applicability in computer graphics/CAD. Lecture hours (2)
- From polynomials to NURBS and back: What mathematical properties are required? Power polynomial, Hermite, Bezier, B-spline and NURBS representations, advantages and disadvantages. Variation diminishing, convex hull and other properties. Degree elevation and knot insertion. Methods for drawing (tessellating) curves and surfaces: tabulation, subdivision, chord estimate, conversion to Power polynomials and forward differencing. Lecture hours (8).
- Data structures: Using curve algorithms on surfaces; data structures including OO techniques for curves and surfaces: NURBS data structure and the design of CGU NURBS library. Lecture hours (2).
- Constructing surfaces: Exact representation of conic sections. Extruded surfaces, ruled surfaces, surfaces of revolution skinning. Lecture hours (2)
- Intersecting and filling: Intersecting curves and surfaces; algorithms for filling curved objects. Lecture hours (2).
- 3D Editing and input: Interactive design issues for 3D editing and input. Lecture hours (2)
- Interrogating surfaces: How to tell if a surface is smooth; isoparametric lines, contours, curvature and surface normal plots; orthotonics, isophotes, reflection lines, and focal surfaces.

26 CS634: Electronic Commerce Technologies

Level:	MSc
Credit Rating:	15
Degrees:	ACS, CS
Pre-requisites:	CS533 recommended, although not required. Some knowledge of data modelling and database technologies helpful.
Pre-course work:	40 hours: preparatory reading
Taught week:	40 hours: 40% lectures, 60% supervised lab
Post-course work:	40 hours: unsupervised lab
Assessment:	30% exam, 20% lab, 50% coursework
Lecturer(s):	Dr RW Giordano (richardg@MIT.EDU)

Introduction

Although there are books and short courses on electronic commerce, they take either a business perspective (understanding markets, capturing customer interactions to inform marketing and product-development decisions, strategies on differentiation, etc.) or a surface technical perspective (how to use this tool or that tool and become a millionaire). This course is designed for people who will become IT leaders, and who want to become familiar with underlying internet commerce technologies, particularly strategies of design and choice of technologies. The course unit is essentially an overview of advanced web-based technologies, but with the following emphases: the process of designing advanced web-based systems that serve communities of users; using database technologies to support web-based interactions; and modelling data and processes.

Aims

This course unit provides students with an intensive survey of technologies used to support all aspects of electronic commerce, and to help students see how technologies, tools, and strategies learnt in other CS course units can be applied to internet commerce applications. The overall aim is to develop a familiarity with the concepts and tools of electronic commerce, and to understand the process by which ecommerce systems are designed, implemented, managed, and evaluated. Although students will be exposed to some technologies and strategies specific to internet commerce applications, the intention is that students will understand how to put together what they already know from the CS and ACS course units that have taken at Manchester to build advanced web applications. Because the subject of electronic commerce and its associated technologies is so broad, the course unit itself will be something of an intensive overview. Students will have the opportunity to study in detail one or two aspects of eCommerce technologies through either an individual or joint project. The project will enable the student to gain practical experience by, for example, applying technologies to an ecommerce application, study in detail the technical features of an ecommerce site, investigate markup languages and their semantics in ecommerce contexts. The lectures supplement course and lab work. Students will form into teams, and all laboratory and classroom work will be done by teams. This is done not only to help students learn from each other, but also to give them some real experience in teamwork and team management. Technologies in detail will be described by example to teams.

Learning Outcomes

A student completing this course unit should:

1) have an understanding of how ecommerce and web based applications are desigened, built and implemented. (A)
2) have a knowledge of tools, technologies, concepts and processes, that comprise the technical infrastructure of eCommerce sites and be able to solve problems about site design, hardware and software architecture, and document architecture. (A and B)

3) have a knowledge of data architecture and be able to solve problems about modelling data and processes so that they can be discovered in web-based environments (A and B)
4) be able to design an ecommerce or advanced web application and evaluate and justify the design. (B)
5) be able to encode data in XML and prepare a technical report on the modelling and ontologies as they relate to a web site in question. (C)
6) be able to work effectively as a member of a group to design and implement a web-based application in a real-world environment. (D)

Assessment of learning outcomes

Learning outcomes (1), (2) and (3) are assessed by examination,
learning outcome (4) by examination and in the laboratory and
learning outcomes (5) and (6) in the laboratory

Contribution to programme learning

A2, B3, C1, C4, D1, D2, D4.

Reading list and supporting material

Philip Greenspun. Philip and Alex's Guide to Web Publishing (San Francisco; Morgan Kaufman, 1999) ISBN: 1-55860-534-7. The lecturer produces web page with links to web-based readings for each of the lecture topics.

Special resources needed to complete the course unit

Students are expected to build a working prototype web-site during the week of lectures, and to refine the site during the week following the lectures. Students form into teams, and it is important that the teams remain intact during the week following the lectures. Moreover, it is important that all materials and work completed by students are loaded on CS School hardware, and that student have access to the CS web server.

The number of students on this course unit is restricted to 70.

Detailed syllabus

Introduction [1]. Why we are here. Course administration. How to choose a good problem. The responsibilities of the software engineer (that is, software engineering for cavemen).

The sociology and psychology of electronic communties [2]. Building, recognizing, managing and making use of online communities in web-based environments (such as communities of practice, communities of purpose). Theories of online presence and cooperation.

A Guide to eCommerce in General [2]. How to differentiate eCommerce today from eCommerce yesterday. Current problems of eCommerce. Interesting solutions and approaches to those problems.

A Guide to Knowledge Commerce [1]. Understanding knowledge as a commodity and as a process, and representing it in web-based environments.

Web architecture. [3]. Structural design of eCommerce systems. Client-server architecture, 2-, 3-,n-tier design, server farms, scalability. Integration of legacy systems. Java Beans, Enterprise Java Beans (EJB), Java Server Pages (JSP). Particular problems posed by 24/7operation and an open user community.

Data interchange [3]. Exchanging data over the Internet. XML, style sheets, document type definition (DTD); metadata and document discovery. Interchange of processes using WSDL and SOAP (as examples).

Usability [2]. User-interface design for web-sites. Use of HCI methodologies in evaluating user interfaces.

Electronic payments [1]. technologies that support the processing of electronic payments. Characteristics and properties of electronic payment systems. Formalisms of correctness.

Mass personalization and the virtual customer [1]. Automation of the customer relationship. Use of data to customize the web experience. Cookies and their risks. Obtaining and using personal information. Rule-based filtering, implicit profiling, collaborative filtering.

27 CS635: Decision Analysis and Decision Support Systems

Level:	MSc
Credit Rating:	15
Degrees:	ACS/CS
Pre-requisites:	None
Pre-course work:	40 hours: preparatory reading including preparation of a presentation on given paper
Taught week:	40 hours: lectures, seminars and laboratories
Post-course work:	40 hours: project work based upon the student's preferences (software development, case study, academic study, etc.)
Assessment:	33% exam, 17% coursework essay, 50% project
Lecturer(s):	Prof. Simon French and Dr. Nadia Papamichail
Limit on numbers:	50 participants

Introduction

The course looks at decision making, in particular how people do and should make decisions. It uses this understanding to discuss decision support systems and decision analytic software.

Aims

The course aims to provide students with an overview of various decision support, operational research and artificial intelligence systems and the ways in which they support effective decision making in organisations. It will discuyss cognitive biases in decision making and how these may be countered through decision support techniques. It will also draw upon some of the normative theories of how people should make decisions. Finally it will consider several examples of decision support suystems, including one in depth case study, to explore how theory and practice come together in implementation.

Learning Outcomes

A student completing this course unit should:

1) have a multi-disciplinary understanding of behavioural and normative theories of decision making, the value to individuals and organisations of decision support systems and be aware of current practice in the use of decision support systems. (A)
2) have a knowledge of decision analytic techniques and be able to solve some simple decision problems. (A and B)
3) be able to design (in outline) decision support systems and processes, evaluate and justify the design. (B)
4) be able to evaluate the appropriateness of decision support systems in various parts of organisations and prepare a presentation on their conclusions (B and C)
5) be able to work effectively as a member of a group to evaluate decision support systems and also to analyse decision problems more generally. progspec(D)

Assessment of learning outcomes

Learning outcomes (1), (2) and (4) are assessed by examination,
learning outcome (1), (4) by coursework essay
learning outcomes (1), (2), (3), (4) and (5) by project

Contribution to programme learning

A2, B2, B3, C2, C4, D1, D2, D3, D4.

Reading list and supporting material

P.R. Kleindorfer, H.C. Kunreuther, P.J.H. Schoemaker 'Decision Sciences: an integration perspective' Cambridge University Press 1993

G.M. Marakas, Decision Support Systems in the 21st Century, Prentice Hall, 1999.

E. Turban and J.E. Aronson (2001) Decision Support Systems and Intelligent Systems. 6th Edition. Prentice Hall.

An extensivce website is provided with notes and other materials including web links.

Detailed syllabus

Overview of different types of decision making: strategic, tactical and operational. Consideration of organisational structures. Mapping of databases, MIS, EIS, KBS, expert systems, OR modelling systems and simulation, decision analytic systems onto activities within an organisation. Extension to other 'non organisational' areas of decision making, e.g. military and emergency management.

Studies of human cognition in relation to decision making and the assimilation of information. Cultural issues. Implications for design of decision making support. Communication issues.

Normative, descriptive and prescriptive analysis: requisite modelling. Contrast with recognition primed decision tools.

Database, MIS, EIS, KBS, Belief nets, data mining. OR modelling tools: simulation and optimisation. History, design, implementation: benefits and pitfalls. Risk assessment. Decision analysis and strategic decision support. Group decision support systems and decision conferencing. Intelligent decision support systems: tools and applications. Cutting-edge decision support technologies.History, design, implementation: benefits and pitfalls.

Quality assurance and validity of decision support.

RODOS: A decision support system for nuclear emergencies. In depth study of a system in which almost all of the techniques come together into one system. Discussion of design. Implementation issues.

28 CS636: Advanced Database Technologies

Level:	MSc
Credit Rating:	15
Degrees:	ACS
Pre-requisites:	Good familiarity with relational databases and programming.
Pre-course work:	40 hours: Preparatory reading and introductory lab
Taught week:	40 hours: 50 % lectures, 50 % supervised lab
Post-course work:	40 hours: Unsupervised lab
Assessment:	33 % exam, 67 % coursework
Lecturer(s):	Dr Ulrike Sattler and Dr Alvaro A. A. Fernandes
Course Unit webpage:	http://www.cs.man.ac.uk/~sattler/teaching/cs636.html
Limit on numbers:	50 participants

Introduction

This course unit is divided into two parts: one on Data Warehousing and one on Data Mining. Both data warehousing and data mining are advanced recent developments in database technology which aim to address the problem of extracting information from the overwhelmingly large amounts of data which modern societies are capable of amassing. Data warehousing focuses on supporting the analysis of data in a multidimensional way. Data mining focuses on inducing compressed representations of data in the form of descriptive and predictive models.

Aims

The data warehousing part of the course unit aims to give students a good overview of the ideas and the techniques which are behind recent developments in the data warehousing and On Line 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 course unit 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

A student completing this course unit should:

1) have an understanding of the foundations, the design, the maintenance, the evolution, and the use of data warehouses, by looking at these topics in a rigorous way. (A)
2) have mastered the basic range of techniques for creating, controlling, and navigating dimensional business databases, by being able to use a powerful tool for dimensional modelling and analysis. (B, C and D)
3) have an understanding of the data mining process, its motivation, applicability, advantages and pitfalls. (A)
4) have an understanding of the principles, methods, techniques, and tools that underpin successful data mining applications. (A and C)
5) be able to apply the methods and techniques surveyed in the course using a data mining workbench. (B, C and D)

Assessment of learning outcomes

Learning outcomes (1) and (3) are assessed by examination, learning outcome (4) by examination and in the laboratory and learning outcomes (2) and (5) in the laboratory.

Contribution to programme learning

A2, B2, B3, C2, D3, D4

Reading list and supporting material

For the data warehousing part of the course unit:

- M. Jarke, M. Lenzerini, Y. Vassiliou, P. Vassiliadis (ed.), Fundamentals of Data Warehouses, Springer-Verlag, 1999.
- Ralph Kimball, The Data Warehouse Toolkit, Wiley 1996.

For the data mining part of the course unit, the lecture notes are the only obligatory reading material, i.e., there's no need for the students taking the course to buy any book. However, these are some textbooks that a student may wish to consult:

- 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.*
- 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 take a machine-learning viewpoint of data mining. It is also useful in covering data warehouses too, to some extent.*
- 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.*
- 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.*

Special resources and limits on participation

The data warehousing labs will be based on the StarTracker software that accompanies *The Data Warehouse Toolkit* book above.

The data mining labs will be based on the WEKA (Waikato Environment for Knowledge Analysis).

Detailed syllabus

Part I

- **Introduction to Data Warehousing [2]**: Heterogeneous information; the integration problem; the Warehouse Architecture; Data Warehousing; Warehouse DBMS.
- **Aggregations [3]**: SQL and aggregations; aggregation functions; grouping.
- **Data Warehouse Models and OLAP Operations [4]**: 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 [2]**: monitoring; wrappers; integration; data cleaning; data loading; materialised views; warehouse maintenance; OLAP servers; metadata.

Part II

- **Introducing Data Mining** [**1**]: 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** [**1**]: Approach; Objectives; Documents; Structure; Binding to Contexts; Phases, Task, Outputs.
- **Data Mining Inputs and Outputs** [**3**]: 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** [**3**]: One-R; Naïve Bayes Classifier; Decision Trees; Decision Rules; Association Rules; Regression; K-Nearest Neighbour Classifiers.
- **Evaluating Data Mining Results** [**3**]: Issues in Evaluation; Training and Testing Principles; Error Measures, Holdout, Cross Validation; Comparing Algorithms; Taking Costs into Account; Trade-Offs in the Confusion Matrix.

Additional Information

Additional information may be found at the course unit webpage.

29 CS639: Computer Security

Level:	MSc
Credit Rating:	15
Degrees:	All Advanced Masters
Pre-requisites:	None
Pre-course work:	40 hours: Introductory lab and preparatory reading
Taught week:	40 hours: Lectures and group work
Post-course work:	40 hours: Groupwork, and the development of a report.
Assessment:	100% coursework
Lecturers:	Unit manager: Daniel Dresner (National Computing Centre) daniel.dresner@ncc.co.uk; Ning Zha

Introduction

Note: This course unit is developed by Daniel Dresner, Standards Manager at the National Computing Centre[10], an independent research organisation which promotes the effective use of information technology. It is located in Manchester (on Oxford Road, near the University).

Aims

The course unit covers the requirements of system security throughout the system development process from the 'Acquisition Preparation' stage to the 'Disposal Process' stage.

Learning Outcomes

A student successfully completing this course unit should:

1) have a good understanding of how to define system security requirements [A1,A2,B2],
2) be able to prioritise requirements, and match requirements to solutions and countermeasures commensurate with associated risks [B2,C1,D1,D3],
3) have a good understanding of the correlation of business processes to technology in relation to security requirements [A1,A2],
4) be familiar with the relevant industry security standards and the regulation, and their application [C4].

Assessment of learning outcomes

All learning outcomes are assessed by the evaluation of a system security management plan for a case study, which each student prepares. A Report detailing a full management plan will be assessed.

Contribution to programme learning

This contributes to Outcomes A1, A2, B2, C1, C4; and in the groupwork and report preparation D1 and D3.

Reading list and supporting material

The following material supports the course unit:

- National Computing Center Guideline 275, *Desert Island Standards*, February 2003.
- National Computing Center Guideline 269, Managing Risk - a practical guide, July 2002.
- BS 7799 Part 1: Code of practice for information security management.

[10] http://www.ncc.co.uk

Detailed syllabus

1a. The need for information assurance Security breaches. Introduction to business continuity. System lifecycles. Trust.

1b. Introduction to standards Plan-do-check-act lifecycles. Overview of BS 7799 Information security management.

2. Information security management Security policy. Security organisation. Asset classification and control. Personnel security. Physical and environmental security. Communications and operations management. Access control. System development and maintenance. Business Continuity management. Compliance.

3. Risk management

4a. Vulnerabilities Windows, Unix and Open source.

4b. Solutions and countermeasures Intrusion detection/prevention. Firewalls. Anti-virus software. Virtual Private Networks (VPNs).

5. Active security Audits and reviews: Vulnerability scanners, Penetration testing. Inspection.

30 CS643: Machine Learning

Level:	MSc
Credit Rating:	15
Degrees:	ACS/CompSci
Pre-requisites:	None
Pre-course work:	40 hours: 80% introductory lab, 20% preparatory reading
Taught week:	40 hours: 50% lectures, 50% supervised lab
Post-course work:	40 hours: 90% unsupervised lab, 10% supervised lab/clinic
Assessment:	33% exam, 67% coursework (laboratory reports)
Lecturer:	Dr. Magnus Rattray
Limit on numbers:	50 participants

Introduction

Machine learning is concerned with how to automate learning from experience. This is typically accomplished by forming models which to some extent describe or summarise experiences, our data, in a useful way. For example, speech recognition software requires examples of continuous speech and will often form a different model for each different user. In this course a variety of machine learning paradigms and algorithms will be introduced which are appropriate for learning from examples with discrete or continuous-valued attributes. The course has a fairly mathematical content although it is intended to be self-contained.

Aims

This course unit aims to introduce the main algorithms used in machine learning, to introduce the theoretical foundations of machine learning and to provide practical experience of applying machine learning techniques.

Learning Outcomes

A student completing this course unit should:

1) have knowledge and understanding of the principle algorithms used in machine learning, as outlined in the syllabus below (A)
2) have sufficient knowledge of information theory and probability theory to provide a theoretical framework for machine learning (A)
3) be able to apply machine learning algorithms, evaluate their performance and appreciate the practical issues involved in the study of real datasets (C)
4) be able to provide a clear and concise description of testing and benchmarking experiments (D)

Assessment of learning outcomes

Learning outcomes (1) and (2) are assessed by examination,
learning outcomes (1),(3) and (4) are assessed by laboratory reports

Contribution to programme learning

A1, A2, C1, D3, D4

Reading list and supporting material

There is a CS643 web page[11] with further details for the current session.

No single textbook covers the entire course and I will supply handouts and review articles for some material which does not appear in available texts. The following book covers many of the algorithms which we discuss and is the main course textbook.

Mitchell, T. M., "Machine Learning" McGraw-Hill, 1997. Introduction to machine learning, covering a broad range of topics and algorithms.

Additional reading

Bishop, C. M., "Neural Networks for Pattern Recognition" Clarendon Press, 1995. Good introduction to neural networks and related statistical methods. Takes a statistical perspective with emphasis on Bayesian inference.

Ballard, D. H., "An introduction to Natural Computation" MIT Press, 1997. Provides a different perspective, with emphasis on the computational aspects of learning algorithms in relation to computational models the brain. Covers some material on control and hidden Markov models not discussed in Mitchell's book.

Baldi, P., Brunak, S., "Bioinformatics: The Machine Learning Approach" MIT Press, 1998. Covers a number of machine learning applications in biology and provides a good introduction to hidden Markov models, neural networks, learning algorithms and Bayesian inference.

Special resources needed to complete the course unit

The matlab programming environment is used in the laboratory. A number of freely available matlab toolboxes are used.

Detailed syllabus

Introduction to machine learning [1]. Overview of different tasks: classification, regression, clustering, control.

Concept learning, information theory and decision trees [2]. Concept learning (algorithms and limitations), Shannon's entropy, mutual information and gain, ID3 and extensions.

Introduction to probabilistic modelling [2]. Probability distributions and densities, Bayes' rule, maximum likelihood, Bayesian inference.

Unsupervised learning [2]. Clustering (Gaussian mixtures, EM-algorithm, k-means), Dimensionality reduction (PCA).

Non-linear regression and classification [2]. Feed-forward neural networks, support vector machines.

Sequence learning [2]. Markov chains, hidden Markov models.

[11] http://www.cs.man.ac.uk/ magnus/CS643/CS643.html

31 CS644: Advanced Machine Vision

Level:	MSc
Credit Rating:	15
Degrees:	ACS
Pre-requisites:	Mathematics, C or Matlab programming
Pre-course work:	40 hours: web directed reading and software projects
Taught week:	40 hours: lectures, group exercises, supervised labs
Post-course work:	40 hours: unsupervised labs, essay
Assessment:	40% exam, 30% lab, 30% coursework
Lecturer(s):	Dr. N.A.Thacker, Prof. C.J.Taylor
Limit on numbers:	50 participants

Introduction

This unit will give students a foundation in the subject of machine vision. This will involve gaining familiarity with algorithms for low-level and intermediate-level processing and considering the organisation of practical systems. Particular emphasis will be placed on the importance of representation in making explicit prior knowledge, control strategy and interpretting hypotheses. This unit will also give students a foundation in the statistical methods of image analysis. This will involve gaining familiarity with probability theory, its simple forms and their limitations. Particular emphasis will be placed on the importance of understanding algorithmic stability and optimality as a framework for algorithmic design and research methodology.

Aims

To introduce the basic concepts and algorithmic tools of computer vision with emphasis on industrial and medical applcations.
To introduce the problems of building practical vision systems.
To explore the role of representation and inference.
To explore the statistical processes of image understanding and develop an understanding of advanced concepts and algorithms.
To discuss novel approaches to designing vision systems that learn.
To develop skills in evaluation of algorithms for the purposes of understanding research publications in this area.

Learning Outcomes

A student completing this course unit should:

1) have an understanding of commmon machine vision algorithms. (A)
2) have a knowledge of the statistical design of algorithms. (A and B)
3) have a knowledge of the properties of image data and be able to solve problems about extraction of features and other quantitative information. (A and B)
4) be able to design basic systems for image analysis and evaluate and justify the design. (B)
5) be able to write a program for the analysis of image data and prepare a technical report on the evaluation of this program on suitable test data. (C)
6) be able to work effectively as a member of a group to prepare presenations describing complex machine vision algorithms to their peers. (D)

Assessment of learning outcomes

Learning outcomes (1), (2) and (3) are assessed by examination,
learning outcome (4) and (5) by examination and in the laboratory and
learning outcomes (6) in tutorials.

Contribution to programme learning

This course contributes to learning outcomes; A1, A2, B3, C2, D2.

Reading list and supporting material

All supporting material and the directed reading list can be found at;
http://www.niac.man.ac.uk/Tina/docs/cvmsc/

Special resources needed to complete the course unit

The course requires access to a MATLAB toolkit including image processing course units and access to a suitable environment for web access and programming.

Detailed syllabus

Pre Course

WWW based directed reading guide (see above).
Practicals and assessed work.
Printed tutorials.
Contact week exercises.

Post Course Materials

Essay

Contact Week

Day 1

9.00 Introduction and Review of Precourse Material (CJT,NAT) Including Timetable, Assessment Details, Handouts for CJT's lectures and statistics tutorial.

10.50 Basic Image Analysis (CJT)

13.30 Image Segmentation Revisited (CJT)

14.30 Edge Based Vision (CJT)

15.30 Shape (CJT)

17.00 Summary (CJT) Including copies of journal papers for assessed essay and Statisitics handout.

Day 2

9.00 Demonstrations (Research at WIAU) (AJL)

11.45 Industrial Applications (PC)

13.30 Introduction to Model Based Vision (CJT)

14.25 Preparation for Student Talks (CJT) Including core materials for model based vison presentations.

Day 3

9.00 Deformable Models (CJT)

10.10 Stereo (CJT)

11.10 Motion (CJT)

11.50 Pattern Recognition (CJT)

13.30 Discussion and Exercises (NAT) Including notes for NAT's lectures.

14.25 Statistics and Error Propagation (NAT)

15.20 Assessed Laboratory Exercise (Stereo Vision) (NAT,AJL,MP,CJT)

Day 4

9.00 Stability of Image Processing Algorithms (AJL)

9.55 Statistical Foundations of Algorithmic Design. (NAT)

10.50 Neural Networks in Machine Vision (NAT)

11.45 Exercises (NAT) Hand outs for practical exercises.

13.30 Data Fusion (NAT)

14.25 Image Warping (NAT)

15.20 Students Deliver Prepared Talks (CJT,NAT)

Day 5

9.00 Colour and Texture (PB)

9.55 Advanced Linear Operators (NAT)

10.50 Wavelets (NAT)

11.45 Laboratory Exercise (Wavelet Compression)

13.30 Corner Detectors (NAT)

14.25 Stereo Geometry and calibration (NAT)

15.20 Demonstrations: a Robotic Vision System (AJL)

16.15 Completeness Properties (NAT)

32 CS646: The Semantic Web: Ontologies and OWL

Level:	MSc
Credit Rating:	15
Degrees:	Advanced/CS
Pre-requisites:	A knowledge of basic logic
Pre-course work:	40 hours: preparatory reading and exercises
Taught week:	40 hours: 60% lectures, 40% supervised lab
Post-course work:	40 hours: 90% unsupervised lab or assignment
Assessment:	40% exam, 30% Taught Week Labs and Exercises, 30% Post-course work
Lecturer:	Prof. Ian Horrocks, Prof. Alan Rector.
Limit on numbers:	50 participants

Introduction

Knowledge representation and "ontologies" are becoming critical to the development of the next generation Web ("The Semantic Web" and "meta data"). The course will present the knowledge representation paradigms used in a variety of applications including current research in the school in "E-Science" and theWeb. Describing web resources with metadata expressed using ontologies is a key step towards achieving effective 'agent based' applications to automate web operations.

Aims

The couse will provide studens with a theoretical and practical understanding of leading edge solutions for the Semantic Web and for knowledge representation more generally. It will introduce students to description logics through the the new W3C standard Web Ontology Language, OWL. Much of the development of OWL has taken place at the University of Manchester under Ian Horrocks.

Learning Outcomes

A student successfully completing this course unit should:

1) Be able to discuss/explain the general principals of semantic networks, frames, rules (A),
2) Be able to discuss/explain KR/ontology languages designed for the world wide web, in particular the new Web Ontology Language (OWL) (A, B),
3) Understand the syntax, semantics and decision procedures for the famly of description logics which underpin OWL (A),
4) Know the common ontological structures and principles of ontology development , have an appreciation of "why it's hard", and to be able to write critically about current work on the "Semantic Web" (A, B),
5) Be able to design and build ontologies in OWL using the *de facto* standard editor, OilEd, justify and evaluate their design (B, C), and explain their behaviour.

Assessment of learning outcomes

Learning outcomes 1-4 will be assessed by exam. Learning outcome 5 will be assessed via practical and post-course work.

Contribution to programme learning

A1, A2, B2, B3, C1, C3, D3, D4

Reading list and supporting material

The Description Logic Handbook, Baader et al, CUP, 2003.

Ian Pratt. Artificial Intelligence. Macmillan, 1994.

John Sowa. Principles of Semantic Networks: Explorations in the representation of knowledge. Morgan Kaufmann, 1991.

Russell and Norvig. Artificial Intelligence: A Modern Approach. Prentice Hall, 1995.

Han Reichgelt. Knowledge Representation: An AI Perspective. Ablex Publishing, 1991.

Selected papers and technical reports will be distributed during the lectures. These will include:

- Tim Berners-Lee, James Hendler and Ora Lassila. The Semantic Web. Scientific American, May, 2001.
- Ian Horrocks, Peter F. Patel Schneider, and Frank van Harmelen. Reviewing the design of DAML+OIL: An ontology language for the semantic web. In Proc. of AAAI-2002, 2002.
- S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens. OilEd: a reason-able ontology editor for the semantic web. In Proc. of the Joint German Austrian Conference on AI,
- W. Woods and J. Schmolze. The KL-ONE Family. Computers and Mathematics with Applications, Vol 2-5, pp 133-177, 1992.
- Franz Baader, Ian Horrocks, and Ulrike Sattler. Description logics as ontology languages for the semantic web. In Dieter Hutter and Werner Stephan, editors, *Festschrift in honor of Jorg Siekmann*, Lecture Notes in Artificial Intelligence. Springer, 2003.
- Ian Horrocks and Peter F. Patel-Schneider. Three theses of representation in the semantic web. In *Proc. of the Twelfth International World Wide Web Conference (WWW 2003)*, 2003.

Detailed syllabus

The following lists topics to be covered in the pre-course work and taught week.

Precourse Work

Background reading on knowledge representation, ontologies, and the Semantic Web

Background review of logic

Frames and Semantic Nets

Initial OilEd tutorials (available along with oiled from oiled.man.ac.uk)

Course work

Basics of knowledge representation and informal introduction to OWL

Description logics and classiers - the ALC family and its extensions

Expressiveness versus tractability; highly expressive description logics; implemented description logic systems; description logics and the "Semantic Web".

Practical issues in ontologies: Basic principles, normalisation and the "Ontoclean" methodology, upper ontologies,

Common problems in ontology development: parts and wholes, time, space, fundamental limitations.

Laboratory Work

Introduction to OilEd and OWL including advanced tutorial

Special problems of representation and reasoning in OWL

Differences between 'open world' reasoning in OWL and 'closed world' reasoning in databases and logic programming - "Ontologies for Pizzas"

Postcourse work

Practical development project
- Problem sets
- Critique/comment on implemented ontologies on the Web
- Additional information may be found here.

33 CS648: Neural Networks

Level:	MSc
Credit Rating:	15
Degrees:	ACS/CS/CompSci
Pre-requisites:	None
Pre-course work:	30 hours: 66% introductory exercise, 33% preparatory reading
Taught week:	40 hours: 60% lectures, 40% supervised lab
Post-course work:	50 hours: 50% unsupervised lab, 50% essay
Assessment:	33% exam, 40% laboratory work, 27% essay
Lecturer:	Dr. Jonathan Shapiro
Limit on numbers:	50 participants

Introduction

This course focuses on the foundations of neural network theory and the application of neural network models in engineering, cognitive science, and artificial intelligence. The course will present the major neural network paradigms: attractor neural network models of memory, a sequence of supervised learning models of increasing complexity, a sequence of unsupervised clustering and categorisation networks, reinforcement learning networks, and aspects of learning theory. Practical experience with the models will be integrated into the course. A final computer project and an essay on related work will constitute post-course work.

Aims

Learning Outcomes

A student completing this course unit should:

1) have an understanding of the concepts and techniques of neural networks through the study of the most important neural network models. (A)
2) have a knowledge of sufficient theoretical background to be able to reason about the behaviour of neural networks (A and B)
3) be able to evaluate whether neural networks are appropriate to a particular application (B)
4) be able to apply neural networks to particular applications, and to know what steps to take to improve performance (B)
5) have knowledge of research literature on neural networks in one particular domain, and be able to put new work into context of that literature (C and D)

Assessment of learning outcomes

Learning outcomes (1), (2), (3) and (4) are assessed by examination,
learning outcome (1), (2), and (4) by examination and in the laboratory and
learning outcomes (5) by an essay related to chosen final laboratory project

Contribution to programme learning

A1, A2, B2, B3, C4, D3.

Reading list and supporting material

Haykin, S., Neural Networks - A Comprehensive Foundation (2nd Edition). Macmillan, 1999.

Special resources needed to complete the course unit

Laboratory work will be done in matlab using the Neural Network Toolbox, and public domain toolboxes.

Detailed syllabus

Motivation [1]. The role of neural networks in engineering, artificial intelligence, and cognitive modelling.

Supervised learning in neural networks [4]. Feed-forward neural networks of increasing complexity, gradient descent learning and extensions, learning and generalization theory

Computation and dynamical systems [4] Hopfield model of content-addressable memory, Hopfield-Tank approach to optimisation, resistive networks for vision models, complex dynamical learning models.

Reinforcement Learning [4] The problem of reinforcement learning, Arp learning, Q-learning, TD-learning. Generalization and function approximation.

Unsupervised Learning [4] Competitive learning, Self-organizing feature maps, ART networks, GWR networks.

Selected Applications [8]

34 CS649: Robotics

Level:	MSc
Credit Rating:	15
Degrees:	ACS, CS
Pre-requisites:	None
Pre-course work:	Preparatory reading and on-line test
Taught week:	Lectures and supervised lab
Post-course work:	Robot design and analysis
Assessment:	70% 2h exam, 10% Lab, 10% Post course work, 10% Pre-course work on-line internet test
Lecturer:	Dr Robert Richardson
Limit on numbers:	50 participants

Aims

This course unit introduces students to robotic systems coving multi-link robotic systems, mobile robotic systems, actuators, sensors, biologically inspired robotics and machine learning techniques. The main aim is to give students an introduction to the field, historic background, development and current cutting edge research points, as well as a practical introduction how to move and control robots. The course unit is practical, and students will be given access to robots for exercises.

Learning Outcomes

At the end of the course unit students will be able to:

(1) Describe different mechanical configurations for robot manipulators
(2) Have an understanding of the functionality and limitations of robot actuators and sensors
(3) Undertake kinematic analysis of robot manipulators
(4) Understand the importance of robot dynamics
(5) Understand and be able to apply a variety of techniques to solve problems in areas such as robot control and navigation
(6) To be able to program a robot to perform a specified task
(7) Understand how simulations of robots work, where they can be useful and where they can break down.
(8) Appreciate the current state and potential for robotics in new application areas.

Assessment of learning outcomes

Understanding of the topics covered in the course is assessed in two ways. A 2 hour examination covers the students understanding of the theoretical issues, such as robot control paradigms, machine learning techniques, actuator and sensor theory. The ability to use this knowledge in a practical manner is tested through practical sessions with robots. Practical sessions are marked by the lab demonstrators.

Contribution to programme learning outcomes

The course contributes towards knowledge and understanding of Computer Science through its practical orientation towards programming robots, signal processing in real time, controller architecture and hardware issues. Intellectual skills are trained through the analysis of control problems, identification of ways of solving them and implementation of the solution. Successes or failures are immediately evident through the resulting robot behavior. Practical skills are trained through the practical sessions of the course. Finally transferable skills are trained by having to work tight (lab

session) deadlines working in groups during practical sessions, understanding task statements, analyzing them and solving problems.

Skills include: A1, A2, B2, B3, C1, D1, D4

Reading list and supporting material

There is no set text for this course, and the lecture notes aim to be self-contained. The following books provide useful supporting material for certain sections of the course.

- Phillip McKerrow, Introduction to robotics, Addison-Wesley 1991.
- Robin Murphy, Introduction to AI robotics, MIT Press 2000.

Special resources needed to complete the course unit

Students will use the Robotics Laboratory. The number of students on this course unit is limited to 20.

Syllabus

- Introduction
  Definitions and history of robotics.
- Sensors and actuators
  Types of actuator, types of sensor.
- Robotic systems
  Robot design, biologically inspired robotics, kinematics, dynamics, locomotion, control.
- Autonomous mobile robotic systems
  Benefits, problems, suitable tasks, machine learning, navigation.
- Simulation
  Simulation of a robot and its environment. Assessment of simulation accuracy. Model acquisition and validation.

35 CS699: Research and Professional Skills

Level:	MSc
Credit Rating:	None
Degrees:	All degree programmes except MEnt
Pre-requisites:	None
Pre-course work:	None
Taught week:	40 hours: Mixed activities - lectures, seminars, group skill activities
Post-course work:	None
Assessment:	None
Lecturers:	Various Contributors: Including the Careers Service, the Post-Experience Vocational Education Unit, Course Directors, Research Staff and Groups, and Industrial Consultants

Introduction

This course unit covers material that is presented at various points through the academic year. Part of the course unit provides training in research skills and an orientation towards the practice of research. The other part provides training in a range of professional skills and material on expectations and conduct in an industrial and business environment.

It is presented by a range of staff both internal and external, including the Careers Service, Programme Directors, Academic Staff, Research Staff and Groups, Industrial Consultants, and representatives from a Professional Society.

Aims

This course unit has two aims:

(1) Most of the course unit takes place before students begin work on the research project. It offers 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 course unit.

(2) The course unit also covers various aspects of Professional Skills as required in the IT industry and in Research and Development. There is a presentation on professional ethics and workplace conduct given by a representative of the British Computer Society. The skills necessary in the IT industry are taught through the Careers Service and external consultants from the IT industry. 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

At the end of the course unit the student will:

- be prepared to undertake the Research Project, having been introduced to the skills and knowledge necessary to undertake the project (B and C),
- have presented a research seminar to an audience of researchers (D),
- have been prepared for some of the demands of, and skills required for, work in IT and IT-related industries (A).

Assessment of learning outcomes

There is no formal assessment for this course unit, but active participation is required, and students will need some of the material to succeed in the Research Project. The research seminar is assessed to provide feedback on performance both in the seminar and in the project to date.

Contribution to programme learning

A1 Knowledge and understanding of professional issues.
B1 Introduction to developing original ideas in a research context.
B3 Introduction to problem solving skills in an academic and industrial context.
C2 Training in organising a scientific or industrial research project.
D2 The preparation and presentation of seminars to a professional standard.
D3 Introduction to the preparation of theses and reports to a professional standard.
D4 Introduction to time-management for research projects.

Reading list and supporting material

Lecture notes will be provided and guidance on suitable literature.

Detailed syllabus

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
2. Professional Skills
   - Professional ethics and conduct
   - Professional skills: Including teamwork skills, industrial problem-solving, leadership skills and communication and presentation skills. Job applications, careers advice and interview skills.

Additional information and supporting material for this course unit is available here[12].

[12] http://www.cs.man.ac.uk/Postgrad/ACS-CS/webpages/CS699/profissues.html

36 CS851: The IT Change Agenda

Aims of course unit

The course is designed to equip leaders of IT/Organizational change projects with insights, theory and know-how.

It is taught by Duncan Shaw (d.shaw@mbs.ac.uk) and Peter Kawalek (pkawalek@mbs.ac.uk) of Manchester Business School.

The course unit will be limited to 50 participants.

Learning outcomes

By the end of the course, participants will:

- Have a better understanding of the likely impacts of IT on organizational change.
- Be conversant with models and theories to explain the prospects.
- Be conversant with e-business concepts and theory.

Syllabus and timetable

The timetable of the taught week and the schedule of topics is available here[13].

References

1. Butler, P., A Revolution in Interaction, The McKinsey Quarterly, 1997, Number 1.
2. * Business Week, Rethinking the Internet, 26th March, 2001.
3. Cabinet Office, e.gov Electronic services for the 21st Century, Performance and Innovation Unit, September 2000.
4. * Downes, L., Mui, C., Unleashing the Killer App., Harvard Business School Press, Boston, 1998
5. * Christensen, C.M., The Innovators Dilemma: When New Technologies Cause Great Firms to Fail, Harvard Business School Press, Boston, 1997.
6. Christensen, C.M., The Rules of Innovation, Technology Review, June 2002. www.technologyreview.com
7. Clegg. S. R., Frameworks of Power, Sage Books, London, 1989.
8. Davenport T.H. (1993) - Process Innovation: reengineering work through information technology.
9. * Downes, L., Mui, C. Unleashing the Killer App., Harvard Business School Press, Boston, 1998.
10. Financial Times (2000) Life on the Net, www.ft.com/lifeonthenet
11. Kalakota, R., Whinston, A.B., (1997), Electronic Commerce: A Manager's Guide, Addison Wesley Longman.
12. * Lacity M.C., Willcocks, L.P. (2001) Global Information Technology Outsourcing, John Wiley and Sons.
13. * Moss-Kanter, R., Evolve, Succeeding in the Digital Culture of Tomorrow, Harvard Business School Press, Boston, 2001.
14. Ould, MA., (1995) Business Processes: Modelling and Analysis for Reengineering and Improvement, John Wiley and Sons.
15. Osborne, Gaebler (1993) Reinventing Government.

*denotes key text

[13] http://www.cs.man.ac.uk/Postgrad/webpages/syllabus/docs/CS851.doc

Assessment

- 60% individual assignment based upon a literature survey related to business strategies.
- 40% group project, based on a given problem.

Individual assignment

You should research one of the companies given below and answer the following questions:

1. What are the main components of the business strategy of the company and how do information systems relate to the business strategy?

2. How does the business and information systems strategy relate to the models and theories given in class?

Choose a company from the following list:

* Amazon * Covisint * Egg * EzGov * General Electric * Overture.com * Motorola * Smile Bank * Tesco

Use Google and other search engines to compile your evidence. Use the Financial Times (including ft.com), Hoovers.com, Business Week, The Economist and any other sources you find to be useful. Make sure you reference all your sources. Your final report should be in the region of 2,500 words.

Group Project

1. Student entrepreneurship
2. The school of the future
3. Local Government Outsourcing.

Details will be given in class.

Pre-Course Reading

Please prepare for your individual assignment by undertaking research and reading from the reference list given above.

37 CS852: IT Systems and Strategy

This course unit will explore information systems (IS) technologies and their relationship in supporting and shaping an organisation's strategy.

It is taught by Dr. Nadia Papamichail (n.papamichail@mbs.ac.uk) and Dr. Saleema Kauser (s.kauser@mbs.ac.uk) of Manchester Business School.

The course unit will be limited to 50 participants.

Aims of Course Unit

The course unit will seek to demonstrate how IS technologies can help shape and achieve an organisation's strategic objectives. Specifically, the aims of the course unit are:

- to review a wide range of IS technologies including knowledge management and e-business systems and their effect on business strategy and operations.
- to explore strategic analysis, decision-making and strategy implementation from the point of view of the general manager for any type of organisation.
- to develop skills and knowledge, which can be applied to understanding, formulating and implementing a business strategy.
- to evaluate the different models that have been proposed to explain and assist strategic thinking.
- to analyse the relationship between business and I S strategies and the formulation of effective IS strategies.

Learning Outcomes

The student will gain:

- an understanding of the business and organisation imperatives deriving from IS and e-business systems;
- an awareness of the need to fit systems to strategic objectives and vice versa;
- an overview of the functionalities of many of the systems currently available.
- an understanding of the benefits for organisations to have a strategic focus; be able to apply techniques of industry analysis, environmental analysis and competitor analysis so that they can contribute to the strategic management of an organisation; and understand how managers implement strategy in organisations.

Syllabus

- Strategy: Introduction to basic concepts of strategy, strategic planning. Development of mission, objectives and strategy.
- An analysis of organizational and environmental situations in order to implement strategic plans.
- Competitive models for business strategy; concept of generic strategies; competitive advantage and strategic implementation.
- Information systems: databases, information and knowledge management. Dealing with legacy systems. Fit with organisational process, structure and culture.
- The new e-world: e-commerce, e-business, e-markets, e-government. Current trends and the business imperatives arising from these.
- Fitting information and e-business systems to an organisation's strategy and vice versa.

References

K.C. Laudon and J.P. Laudon (2001) Management Information Systems. 7th Edition. Prentice Hall, Upper Saddle River, New Jersey.
Robert M. Grant, Contemporary Strategic Analysis - Concepts, Techniques and Applications, Basil Blackwell, 1987, 3rd Edition.

Assessment

- 30% group presentation based on a case study provided prior to the course.
- 70% case study examination.

38 Course Units via E-Learning

There is a range of course units taught only via e-learning, provided by the PEVE Unit[14] in the School of Computer Science. These are at various postgraduate levels and credit ratings. Some are already available, others are in preparation.

Currently the following distance learning courses are on offer.

- CS804: Low-Power System Design (Advanced course unit)
- CS810: Programming in C (Foundation course unit)
- CS811: Object-Oriented Programming using Java (Foundation course unit)
- CS818: Object-Oriented Analysis & Design with UML (Advanced course unit)
- CS821: Self Timed Logic (Asynchronous Design) (Advanced course unit)

As a general guide the start dates are October or March each year, but exact dates are to be confirmed. Applications to take these course units are made via the PEVE Unit. Fees may be payable.

For details of these courses, dates and how to apply, see the web page[15] or contact the PEVE Secretary (peve@cs.man.ac.uk).

In order to take any of these course units as part of a university degree programme, the course unit must be appropriate for the degree and the Programme Director for the degree programme must give authorisation. A Modular Masters course is also available combining e-learning course units with face-to-face teaching. See the web page[16] for details of this approach to MSc courses.

[14] http://www.cs.man.ac.uk/peve/peveWebNew/home/index.htm
[15] http://www.cs.man.ac.uk/peve/peveWebNew/e_learning/index.htm
[16] http://www.cs.man.ac.uk/peve/peveWebNew/education/index.htm