The University of Manchester
Faculty of Engineering and Physical Sciences
School of Computer Science

MSc in Computer Science

including the

Diploma and Certificate in Computer Science

Syllabus

2004/2005

Department of Computer Science
Faculty of Science and Engineering
The Victoria University of Manchester

Version of February 11, 2005

URL: `http://www.cs.manchester.ac.uk/Study_subweb/Postgrad/ACS-CS/webpages/syllabus/cs/`

This document describes the syllabus and aims and learning outcomes of the MSc in Computer Science programme for the current session. The (advanced) 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 However, various factors may cause a course unit to be withdrawn (including availability of staff and demand from students) and so there is no guarantee that any particular course unit will be available, though as a School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.

Contents

1 Aims and Learning Outcomes of the Programme 4

1.1 For the MSc Degree 4

1.2 For the Diploma 5

1.3 For the Certificate 6

2 CS500: Introduction to Computer Science 8

3 CS501: Machine Architecture 10

4 CS503: System Software 12

5 CS510: An Introduction to Programming Using C 14

6 CS511: Object-oriented programming in Java 16

7 CS530: Database Architecture Models and Design 18

8 CS533: Information Systems Development 20

9 CS602: Grid Computing and eScience 24

10 CS616: Knowledge Representation and Reasoning 27

11 CS618: Object-Oriented Analysis and Design (for CS students) 30

12 CS619: Extreme Java 33

13 CS624: Mobile Computing 36

14 CS631: Computational Biology 38

15 CS634: Electronic Commerce Technologies 40

16 CS635: Decision Analysis and Decision Support Systems 43

17 CS644: Advanced Machine Vision 45

18 CS648: Neural Networks 48

19 CS649: Robotics 50

20 CS699: Research and Professional Skills 52

21 Course Units via E-Learning 54

1 Aims and Learning Outcomes of the Programme

1.1 For the MSc Degree

Programme Aims

The programme aims to:

- impart a sound understanding of the general principles of Computer Science to students who have a strong academic track record in science, engineering, or other numerate discipline, but who have little formal training in computing;
- provide an in-depth treatment of selected, leading-edge, research-based topics within computing;
- provide sufficient breadth and depth of experience in up-to-date methodologies to significantly advance the career prospects of students within the IT industry, and/or equip them to undertake research in computer science;
- equip students with knowledge to enable them to develop and apply new computer technologies in their original discipline.

Programme Learning Outcomes

A. Knowledge and Understanding of
   1. fundamental topics in Computer Science, including hardware and software architectures, software engineering principles and methodologies, operating systems and software tools;
   2. selected advanced topics to provide a deeper understanding of some aspects of the subject, such as hardware systems design, object-oriented analysis and design, e-commerce technologies, and artificial intelligence;
   3. research methodology and practice;
   4. personal responsibilities and professional codes of conduct.

B. Intellectual (thinking) skills – able to
   1. perform problem analysis from written descriptions;
   2. derive requirements specifications from an understanding of problems (analysis, synthesis);
   3. create and/or justify designs to satisfy given requirements (synthesis, evaluation, application);
   4. develop original ideas in a research context (synthesis).

C. Practical skills – able to
   1. effectively apply software systems and tools under Unix/Windows, underpinned by a knowledge of how those systems work;
   2. implement hardware and/or software designs to provide working solutions, including use of appropriate programming languages, web-based systems and tools, design methodologies, and database systems.

D. Transferable skills – able to
   1. communicate effectively by oral, written and visual means;
   2. use IT skills and display mature computer literacy;
   3. work effectively as an individual and as a member of a team;
   4. perform independent and efficient time management;
   5. plan and manage an individual research project of significant size;
   6. perform independent information acquisition and management, using the scientific literature and Web sources;
   7. prepare technical reports, and a dissertation, to a professional standard;
   8. prepare and present seminars to a professional standard.

1.2 For the Diploma

Programme Aims

The programme aims to:

- impart a sound understanding of the general principles of Computer Science to students who have a strong academic track record in science, engineering, or other numerate discipline, but who have little formal training in computing;
- provide exposure to selected, leading-edge, research-based topics within computing;
- provide sufficient breadth and depth of experience in up-to-date methodologies to advance the career prospects of students within the IT industry;
- equip students with knowledge to enable them to develop and apply new computer technologies in their original discipline.

Programme Learning Outcomes

A. Knowledge and Understanding of

   1. fundamental topics in Computer Science, including hardware and software architectures, software engineering principles and methodologies, operating systems and software tools;
   2. selected advanced topics to provide a deeper understanding of some aspects of the subject, such as hardware systems design, object-oriented analysis and design, e-commerce technologies, and artificial intelligence;
   3. basic research methodology and practice;
   4. personal responsibilities and professional codes of conduct.

B. Intellectual (thinking) skills – able to

   1. perform problem analysis from written descriptions;
   2. derive requirements specifications from an understanding of problems (analysis, synthesis);
   3. create and/or justify designs to satisfy given requirements (synthesis, evaluation, application);
   4. explore ideas in a research context (synthesis).

C. Practical skills – able to

   1. effectively apply software systems and tools under Unix/Windows, underpinned by a knowledge of how those systems work;
   2. implement hardware and/or software designs to provide working solutions, including use of appropriate programming languages, web-based systems and tools, design methodologies, and database systems.

D. Transferable skills – able to

   1. communicate effectively by oral, written and visual means;
   2. use IT skills and display mature computer literacy;
   3. work effectively as an individual and as a member of a team;
   4. perform independent and efficient time management;
   5. plan and manage an individual project;
   6. perform independent information acquisition and management, using the scientific literature and Web sources;
   7. prepare technical reports, and a dissertation, to a professional standard;
   8. prepare and present seminars to a professional standard.

1.3 For the Certificate

Programme Aims

The programme aims to:

- impart a sound understanding of the general principles of Computer Science to students who have a strong academic track record in science, engineering, or other numerate discipline, but who have little formal training in computing;
- provide sufficient breadth and depth of experience in up-to-date methodologies to advance the career prospects of students within the IT industry;
- equip students with knowledge to enable them to develop and apply new computer technologies in their original discipline.

Programme Learning Outcomes

A. Knowledge and Understanding of

   1. fundamental topics in Computer Science, including hardware and software architectures, software engineering principles and methodologies, operating systems and software tools;
   2. personal responsibilities and professional codes of conduct.

B. Intellectual (thinking) skills – able to

   1. perform problem analysis from written descriptions;
   2. derive requirements specifications from an understanding of problems (analysis, synthesis);
   3. create and/or justify designs to satisfy given requirements (synthesis, evaluation, application);

C. Practical skills – able to

   1. effectively apply software systems and tools under Unix/Windows, underpinned by a knowledge of how those systems work;
   2. implement hardware and/or software designs to provide working solutions, including use of appropriate programming languages, web-based systems and tools, design methodologies, and database systems.

D. Transferable skills – able to

   1. communicate effectively by oral, written and visual means;
   2. use IT skills and display mature computer literacy;
   3. work effectively as an individual and as a member of a team;
   4. perform independent and efficient time management;
   5. plan and manage an individual project;
   6. perform independent information acquisition and management, using the scientific literature and Web sources;
   7. prepare reports and/or essays on practical work undertaken.
   8. prepare and present seminars to a professional standard.

Foundation Course Unit Descriptions

2 CS500: Introduction to Computer Science

Level:	MSc
Credit Rating:	None
Degrees:	CS
Pre-requisites:	None
Pre-requisite for:	CS510
Co-requisites:	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 topics in computer science. This course unit provides orientation lectures including an overview to software, an introduction to programming and material on the structure and expectations of the programme. There is an introduction to the computing facilities and other facilities in the school. There is a reception at the School and an induction by the Graduate School.

For each advcanced 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.

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 an introduction to computing and a broad overview of the major issues, themes and topics in 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 two of the advanced course units delivered in semester 2, 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

1) understand the structure of the MSc Programme, what is expected of the student and procedures for the programme (A1)
2) have an overview of basic issues relating to software and programming (A1)
3) have a basic knowledge of the Linux and Windows operating systems and be able to navigate through a file structure, create and delete directories, and create and edit files (A1)
4) be able to write small programs in C (A1, B1)
5) will have gained an overview of general principles and 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. (A1)

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

- Registration, introduction to the course, the School and the University.
- Introduction to software.
- Introduction to programming in C.
- Introductions to individual advanced course units.

Additional information

The timetable can be found in the Programme Handbook.

3 CS501: Machine Architecture

Level:	MSc
Credit Rating:	10
Degrees:	CS
Pre-requisites:	None
Co-requisites:	None
Pre-requisite for:	None
Taught week:	40 hours: 38% lectures, 62% laboratories and private study
Post-course work:	60 hours: coursework and private study
Assessment:	50% exam, 50% lab/coursework
Lecturers:	Dr Doug A. Edwards

Introduction

This course unit introduces both information representation (binary arithmetic, logical operations and so forth), and computer architecture. The course unit serves as a foundation for further studies in Computer Science.

Aims

The course unit is concerned with the fundamental concepts of computer architecture, that is, the internal structure of the computer as seen by the assembly level programmer. It is at this level that hardware and software meet. The ARM processor, currently the processor of choice for embedded mobile systems, is used as an exemplar. Practical experience of the ARM instruction set architecture is gained in the laboratory through exercises which progressively expose features of the instruction set.

Learning Outcomes

A student completing this course unit should:

1) have a knowledge of the significance of bit vectors as characters, numbers or instructions (A1)
2) have a knowledge of binary number representations and binary arithmetic (A1)
3) understand the ARM instruction set and its registers (A1)
4) be able to write programs in assembly language (B3, C2, D4)
5) understand the ARM exception handling mechanism and its support for the requirements of operating systems (A1)
6) understand what differentiates RISC/CISC machines (A1)
7) be able to use a Computer Based Learning Package. (C1)

Assessment of learning outcomes

Learning outcomes (1), (5) and (6) are assessed by examination.
Learning outcomes (4) and (7) are assessed in the laboratory.
Learning outcomes (2) and (3) are assessed by examination and in the laboratory.

Reading list and supporting material

- A. Clements: "The Principles of Computer Hardware", (ISBN 0 19 856453 8) Oxford University Press. 3rd Edition.
  This book covers several 1st year courses and is a good introduction to Computer Architecture and peripherals. Note that the 2nd edition is not suitable as it does not cover the ARM processor.

– S. B. Furber: "ARM system-on-chip architecture", 2nd Edition ISBN 0-201-67519-6 Addison-Wesley.

  This book is a detailed look at the ARM and its variants with a lot of information on its use in embedded systems. A definitive guide written by the man responsible for much of the first design. The book covers the basic material and then examines the latest ARM processors and their embodiment in typical embedded systems.

– CBT Package: there is an online Computer Based Learning Package "ARM System Design" available on the School network. The package allows students to work at their own pace.

Special resources needed to complete the course unit

The ARM toolkit is required for this course unit. A free version downloaded from the ARM website.

Detailed syllabus

Introduction [1 hour]. Overview, motivation, abstraction, processor organization, why study the ARM.

Binary representations [2 hours]. Digital systems, binary notation, binary functions, interpretation of binary symbols. Data representation: logical values, characters, unsigned integer different number bases.

Signed Integer representations: sign & magnitude, 2's complement, arithmetic, overflow.

ARM Development Tool Kit [1 hour].

Data Processing Instrutions [2 hours]. Basic ARM Instruction Set: data-processing, Branch & Branch and Link.

Accessing Memory [3 hours]. Load and Store Instructions Stacks, Load/Store Multiple Instructions, Addressing Modes, Parameter Passing.

Exceptions [4 hours]. Input/Output, Polling, interrupts General Exceptions handling mechanism, privilege mode.

Aspects of ARM implementations [2 hours]. instruction formats, Pipelining, Thumb instructions set.

Floating Point [2 hours]. IEEE 488 standard, coprocessors.

RISC overview [1 hour]. Overview of RISC Architecture Philosophy, n-address architectures.

4 CS503: System Software

Level:	MSc
Credit Rating:	10
Degrees:	CS
Pre-requisites:	CS501, CS510
Co-requisites:	None
Pre-requisite for:	None
Taught week:	40 hours: 60% lectures, 20% supervised lab, 20% private study
Post-course work:	60 hours: coursework and private study
Assessment:	50% exam, 50% lab/coursework
Lecturers:	Dr Alasdair Rawsthorne and Dr Andy Carpenter
Course Unit webpage:	http://mint.cs.man.ac.uk/CS5031

Introduction

This course unit gives a broad view of some of the major tasks of the system software of a computer system, focussing on networking and operating systems. The course unit assumes familarity with the hardware/software interface of a typical system as described in CS501.

Aims

This course unit outlines the answer to the questions:
What has to happen to transfer data between computers or distributed applications?
What is the purpose of an operating system?
How are the major functions of an operating system organised?

Learning Outcomes

A student completing this course unit should:

1) Have an overview of the hardware and software elements of a network. (A1)
2) Appreciate some of the factors that influence the way in which networks are constructed. (A1, B1)
3) Have an overview of the functions and structure of an operating system. (A1)
4) Appreciate the organisation of system services in a modern networked environment. (A1, B1)
5) Be able to write a program to solve a networking problem. (B1, B2, C2, D4)
6) Prepare a technical report on an aspect of operating system design. (D1, D4, D7)

Assessment of learning outcomes

Learning outcomes (1), (2), (3) and (4) are assessed by examination,
learning outcome (5) in the laboratory and
learning outcome (6) by coursework, a 2000 word report.

Reading list and supporting material

Networking:

- Peterson, L.L., and Davie, B.S., Computer Networks: A Systems Approach 2nd edition. Morgan Kaufmann, 2000.
- Tanenbaum, A.S., Computer Networks 3rd edition. Prentice Hall, 1996.
- Stallings, W., Data and Computer Communications 5th edition, Prentice Hall 1997.

Operating systems:

- O'Gorman, J., Operating Systems, MacMillan (Palgrave) 2000, ISBN 0-333-8022288-8 (Course text).
- Tanenbaum A.S. and Woodhall A.S. Operating Systems: Design and Implementation, 2nd Edition, Prentice Hall 1997.

Special resources needed to complete the course unit

Networking coursework uses a library only available for Linux.

Detailed syllabus

Networking. Introduction: Physical elements of a network, protocol layering and reference models. Physical links. Packet switching. Internetworking.

Operating systems.

- Purpose of operating systems. Typical system structure, virtual machines.
- Multiple processes, process and thread scheduling, multiprogramming, mutual exclusion and synchronisation.
- Polling and interrupt based device scheduling; buffering.
- Memory management: Need for and use of virtual memory, segmented and paged systems, performance issues.
- Resource allocation, deadlock management.
- File systems, directory structures.
- Protection and security.
- Client Server computing: The microkernel as message passer. The remote procedure call paradigm. Client server examples such as the Network File System and X windows.

5 CS510: An Introduction to Programming Using C

Level:	MSc
Credit Rating:	10
Degrees:	CS
Pre-requisites:	CS500
Co-requisites:	None
Pre-requisite for:	CS503, CS511
Taught week:	40 hours: 50% lectures, 30% supervised lab, 20% private study
Post-course work:	60 hours: coursework and private study
Assessment:	50% exam, 50% lab/coursework
Lecturers:	Dr Renate Schmidt
Course Unit webpage:	http://www.cs.man.ac.uk/˜schmidt/CS510/

Aims

This course unit will teach the design and implementation of imperative programs using the C programming language. It assumes no previous knowledge of programming.

Learning Outcomes

A student completing this course unit should:

1) have an understanding of the main programming constructs of C (A1)
2) have an understanding of the role of design in the development of programming solutions to problems (A1, B1, B2, B3)
3) have a knowledge of some standard algorithms and data structures in imperative programming and be able to solve problems using lists, trees and recursion (A1)
4) have the competence to write programs in C. (A1, B1, B2, B3, C1, C2, D2, D4)

Assessment of learning outcomes

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

Reading list and supporting material

This course unit will not follow any book exactly but will provide printed notes of all slides used in lectures. The following book is recommended reading material.

– "Practical C Programming", Steve Oualline 1997, 428 pages, O'Reilly (Nutshell Handbook).

Special resources needed to complete the course unit

The practical work requires a computer and a C compiler. The non-assessed laboratory sessions which support the taught week will use Linux PCs and the GNU C compiler (gcc). The GNU C compiler is freely available for all platforms. Students who intend to complete their practical coursework away from the school are advised to have a C compiler installed on their computer before starting the course unit.

Detailed syllabus

Introduction [1]. What is C? Basics of program writing.

The main constructs of C [11]. Variables, data types, expressions, standard input/output, decision and control statements, arrays, strings, functions, variable scope, structures, pointers and file input/output.

Design techniques and modularisation [4]. The role of design in the development of solutions to problems. Top down structured design. Modular design including the use of separate compilation facilities for C.

Algorithms and dynamic data structures [4]. An outline will be given of the role of standard algorithms and dynamic data structures (linked lists, trees, recursion).

Coursework

The coursework consists of three programming assignments of varying difficulty. The weighting of the marks will reflect the importance of the exercises (approximately 10%:20%:20%). Marks will be awarded for working programs which meet the specifications, simple and clear programming style, and internal documentation.

The exam makes up the remaining 50% of the final mark of the course unit.

Additional Information

Additional information may be found at the course unit webpage.

6 CS511: Object-oriented programming in Java

Level:	MSc
Credit Rating:	10
Degrees:	CS
Pre-requisites:	CS510
Co-requisites:	None
Pre-requisite for:	
Taught week:	40 hours: 40% lectures, 60% supervised lab and private study
Post-course work:	60 hours: coursework and private study
Assessment:	50% exam, 50% lab/coursework.
	The exam will, at least partially, be taken online.
Lecturers:	Dr John Sargeant and Dr Kung-Kiu Lau

Aims

This course unit introduces students to object-oriented programming on the assumption that they are already familiar with imperative programming in C. Its main aim is to teach the principles of object-orientation, using Java as a vehicle, rather than attempting to give a complete exposition of all the features of Java.

Learning Outcomes

A student completing this course unit should:

1) Have a knowledge and understanding of the basic concepts of object-orientation, and their realisation in Java (A1)
2) Be able to take a straightforward requirements specification in English and produce a simple object-oriented design. (B1, B3)
3) Be able to draw UML class diagrams representing simple classes and small groups of classes to facilitate 2 above. (B1)
4) Be able to turn such a design into working Java code. (C1, C2, D2, D4)
5) Be able to make effective use of the the Java API documentation (C1)
6) Be able to communicate with other programmers using the basic language of object-orientation. (D1)

Assessment of learning outcomes

Both the examination and the practical will to some extent assess all five outcomes. However, the examination will focus on outcomes (1), (4), and (6), and the practical work on outcomes (2), (3), (4) and (5).

Note: Outcome 4 is critical, and therefore plagiarism of programs will be detected and harshly dealt with.

Reading list and supporting material

A complete set of notes will be provided with the course.

Before the course unit, students are advised to ensure that their C programming is up to scratch. The textbook recommended below includes a multimedia C course on CD-ROM, which may be useful.

After the course unit, the following is recommended as follow-up reference material:

– Thinking in Java, 2nd edition, Bruce Eckel. Prentice Hall 2000, ISBN 0-130-273633-5.

Students who find the course unit difficult may prefer the textbook to be used for the first year undergraduate Java courses.

– An Introduction to Object-oriented Programming with Java, 2nd edition, C. Thomas Wu. McGraw Hill 2001, ISBN 0-07-118195-4.

Special resources needed to complete the course unit

The practical work requires the Java System Development Kit (SDK, formerly the JDK) version 1.2 or later. This is freely available for all common platforms, but part time students intending to do the assessed practial work away from the school are advised to ensure they have a working version before starting the course, to avoid delays. It can be downloaded from www.java.sun.com.

Detailed syllabus

Introduction to OO and Java [2]. Why OO is funadmentally different from imperative programming. What Java is (and isn't). The Hello World program introduced.

Nuts and bolts [1]. Scalars, Strings, and expressions. Simiarlties and differences to C.

Classes from the outside [1]. Their interfaces.

Classes from the inside [1]. Their implementations.

Control structures and arrays [1]. Review and comparison with C.

Classes: other features [1]. Static variables, visibility control, packages.

Inheritance [2]. Extending classes, abstract classes, overriding, polymorphism and dynamic binding.

Exception handling [1]. The mechanisms and how to use them effectively.

Interfaces [1]. (In the Java-specific sense.)

Graphical I/O [2]. Platform-independent graphics and GUIs. The AWT and Swing. Applets. Event handling.

Stream I/O and object serialization [2]. The stream abstraction, useful stream types. The Hello World program explained. Combining streams. Persistance and object serialisation.

Overview of further topics [2]. Security, performance, inner classes, basic design guidelines.

Coursework

The coursework consists of a substantial programming project, done individually, which requires some simple design work. An interim solution is required (for full time students) at the end of the assessed practical week, to ensure that a substantial proportion of the work is done in that week, when help is available. Of the 50 marks, 30 are for functionality and 20 for programming style. The functionality marks are weighted towards the earlier parts of the exercise, so that the final part is effectively an optional challenge.

7 CS530: Database Architecture Models and Design

Level:	MSc
Credit Rating:	10
Degrees:	CS
Pre-requisites:	None
Co-requisites:	None
Pre-requisite for:	None
Taught week:	40 hours: 63% lectures, 37% supervised lab and private study
Post-course work:	60 hours: coursework and private study
Assessment:	50% exam, 50% lab/coursework
Lecturers:	Prof Ian Horrocks and Dr Robert Stevens
Course Unit webpage:	http://www.cs.man.ac.uk/~horrocks/Teaching/cs530/

Introduction

An introduction to the theory and practice of database architecture and design.

Aims

The aim of the course is to provide students with an overview of database management system architectures and environments, an understanding of basic database design and implementation techniques, and practical experience of designing and building a relational database.

Learning Outcomes

A student completing this course unit should:

1) Be able to discuss/explain the importance of data, and the difference between file management and databases. (A1, D1)
2) Be able to discuss/explain the design of database management system architectures and environments (in particular the Ansi-Sparc model). (A1, D1)
3) Be able to discuss/explain the principals of database design. (A1, D1)
4) Be able to discuss, explain and apply conceptual design methodologies, in particular conceptual design using Extended Entity Relationship modelling. (A1, B3, C2, D1, D4)
5) Be able to discuss, explain and apply the relational model and mappings from conceptual designs, in particular normalisation. (A1, B1, C2, D1, D4)
6) Be able to discuss, explain and apply SQL and the Oracle DBMS. (A1, C1, D1, D2, D4)

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 the laboratory.

Reading list and supporting material

– Elmasri R and Navanthe S. B., Fundamentals of Database Systems, 3rd edition, (ISBN 0-201542633), Addison Wesley 1999.
  This is the major course text.

Special resources needed to complete the course unit

Oracle 8 DBMS

Detailed syllabus

Introduction [1]. Importance of data to an organisation, file management vs databases, what is a database, requirements of a database and database management system.

Architectures [4]. Ansi-Sparc model of databases, components of a database management system, schemas, levels of abstraction and mappings, role of the data dictionary, client-server systems, PC based systems, database servers; distributed systems.

Database Design [4]. Design framework, mappings between abstractions, integrity, compromises.

Conceptual Design [8]. Requirement for conceptual design, Extended Entity Relationship model, object-oriented design, UML.

Logical Design [8]. The relational model, normalisation, relational algebra, SQL, the Oracle DBMS, mapping conceptual design to relational, integrity, views, PL/SQL, triggers.

8 CS533: Information Systems Development

Level:	MSc
Credit Rating:	10
Degrees:	CS
Pre-requisites:	None
Co-requisites:	None
Pre-requisite for:	None
Taught week:	45 hours: 44% lectures, 54% coursework
Post-course work:	55 hours: private study
Assessment:	50% exam, 50% lab/coursework
Lecturers:	Dr R. W. Giordano

Introduction

Never has the "seat of the pants" approach been more evident in systems design than it is today. With the growth of the Web, and its ubiquity in almost all aspects of life, anyone with only a rudimentary knowledge of HTML can fancy that they are a "web designer". It is true that a simple web-site does not require much knowledge of an analysis and design methodology. Yet as systems become more complex and are used in complex environments, a methodology is needed to understand and manage that complexity, to assure reliability, to facilitate maintainability, and foster reuse of code.

Systems development methodologies have emerged over the years to address these needs, the most popular of which is Structured Systems Analysis and Design Methdologogy (SSADM). SSADM is a powerful methodology for the analysis and design of large-scale systems with particular strengths in defining functional decompostion, documentation, and ways of organizing work and projects. Having said this, SSADM exhibits distinct weaknesses when applied to object-oriented systems in general, and web-based systems in particular. This course unit, then, will build on the strengths of SSADM, but focus on the object-oriented design of systems in general, and web-based systems in particular. Having said this, most systems design methodologies do not make good use of the techniques and tools used by product designers (people who design everything from washing machines to submarines). This course unit, however, will indeed view systems development as an instance of product development, examine the tools and techniques (such as trade-off analysis, platform specification, sensitivity analysis, etc.) used by designers engaged in the development complex products, and apply them to information systems development.

There are no Laboratory in this course unit in the sense that students sit in front of a workstation and work through a problem. Instead, throughout the course unit students will work through a case example. That example will be modelling of processes, activities and transactions in the Manchester Piccadilly railway station. The object of this exercise is to see Piccadilly as an information and product transaction and processing centre, to model this, and to represent this as a web-site. The class will be divided into teams, and each team will work on a specific process at Piccadilly. The trick is not only to express each process, but to integrate them into a coherent to build a web-site (a virtual Piccadilly or, perhaps in today's parlance, a "Piccadill-e") of this model.

Aims

This course unit provides students with an introduction to the basic tools and techniques which are used to design and implement information systems in organisations, with an emphasis on web-based systems. Students will be involved in developing the model of a fairly complex information system using a Case Study. This approach will enable the student to gain practical experience of applying project management techniques to their work, understanding potential clients and their needs, using tools and techniques of analysis, and making and evaluating decisions regarding the best approach to take in modelling the system under examination. One feature of this course unit is to develop the idea that information systems are "products", and the process of developing them should be thought of as product development activity. It will introduce students to object oriented analysis and design, using UML as a vehicle. It will not attempt to teach object-oriented programming techniques nor be an exhaustive course in UML. Instead, UML will be the means by which systems are modelled and documented, and through which they can be mapped to software.

Learning Outcomes

A student completing this course unit should:

1) have an understanding of the process and tools of analysis and design in general, and object-oriented analysis in design in particular, with particular focus on distributed and web-based environments. (A1)
2) have a knowledge of SSADM and OO concepts and analysis and design techniques and be able to solve problems the analysis, design and documentation of information systems. (A1, B1, B2, B3)
3) have a knowledge and understanding of the requirements and analysis methods of UML and be able to solve problems about the modelling of complex distributed information systems. (A1, B1, B2)
4) be able to evaluate the success of an analysis and design using metrics relevant to the user. (B3, D7)
5) be able to design a simple web-based information system using UML as a vehicle, and evaluate and justify the design. (B3, C1, C2, D2, D7)
6) be able perform Use Case and Sequence Diagram analysis for problems of limited size. (B1, C2)
7) be able to work effectively as a member of a group to use the above skills in analysing and designing an information system (D1, D3, D4, D8)

Assessment of learning outcomes

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

Reading list and supporting material

– Conallen, Jim. 'Building Web Applications With UML'. Addison-Wesley, 1999, ISBN: 0-20-161577-0.
– Stevens & Perdita, Pooley & Rob. Using UML: Software Engineering with Objects and Components (London: Longmann, 1999), ISBN: 0-20-164860-1.
– Greenspun, Philip. 'Philip and Alex's guide to Web Publishing (San Francisco: Morgan Kaufman, 1999), ISBN: 1-55860-534-7.
– Ulrich, KT and SD Eppinger, Product Design and Development (London: McGraw-Hill, 1999), ISBN: 0-07-1169938.
– Marshall, C, Enterprise Modeling with UML: Designing Successful Software Through Business Analysis (Harlow: Addison-Wesley Longman, 1999), ISBN: 0-201-43313-3.
– Schneider, G and JP Winters. Applying Use Cases: A Practical Guide (Harlow: Addison-Wesley Longman, 1999), ISBN: 0-201-30981-5.
– Avison, Fitzgerald, 'Information Systems Development: Methodologies, Tech niques and Tools', (London: Mcgraw Hill, 1995), ISBN 0-07-709233-3.
– Myers, C., ed. 'Professional Awareness in Software Engineering, (Or Should a Programmer Wear a Suit?) (London: McGraw-Hill, 1995), ISBN: 0-07-707837-3.
– Sauer, C. Why Information Systems Fail: A Case Study Approach. (Henley-on-Thames: Alfred Waller, 1993), ISBN: 1-87-247408-X.

Special resources needed to complete the course unit

No software or special resources are needed except an ability to create and publish web pages.

Detailed syllabus

Introduction [1]. Topics. Introduction to the course unit. Outline of the responsibilities of the analyst.

Overview of Design Methodologies and Paradigms [2]. Overview of SSADM and OO Analysis and Design, and their paradigms. A discussion of their strengths, weaknesses, and elements of their paradigms that are shared.

Project Management [2]. An introduction to the major issues attending project definition and management, with an emphasis on the management of groups. A role-play exercise of a design team is included in this lecture.

SSADM [2]. An introduction the Structured Systems Analysis and Design methodology, focusing on Decomposition, Data Flow Diagrams, Structure Charts and their underlying imperative paradigm.

OO Analysis and Design [3]. An introduction the Object Analysis and Design and its underlying declarative paradigm.

Analysis and Design Using UML [4]. Using UML as a OO modeling tool: Overview of syntax, its strengths and weaknesses.

Use Cases and Sequence Diagrams [3]. Consideration of Use Cases and Sequence Diagrams as modeling and documentation tools.

Implementation considerations [1]. A discussion of technical and economic trade offs.

Quality Assurance and Methods of Evaluation [2]. Building on what was learned in the course unit, a discussion of the relationship between system specification and documentation, and documentation and quality assurance. Methods of evaluation, with an emphasis on participant stakeholder evaluation.

Advanced Course Unit Descriptions

9 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

10 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

A2, B1, B2, B3, C1, C2.

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.

11 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

A2, B3, C2, D1, D3, D4.

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.

12 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

A2, B1, B2, B3, C1, C2, D4.

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.)

13 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

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).

14 CS631: Computational 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)

15 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 (r.giordano@mbs.bbk.ac.uk)

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, B1, B3, C1, C2, D2, D3, D4, D7.

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.

16 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, B1, B3, D1, D3, D6, D8.

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.

17 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)

18 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

A2, B1, B2, B3, D4, D7, D6.

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]

19 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.

20 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[3].

[3]http://www.cs.manchester.ac.uk/Postgrad/ACS-CS/CS699/profissues.php

21 Course Units via E-Learning

There is a range of course units taught only via e-learning, provided by the PEVE Unit[4] 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[5] 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[6] for details of this approach to MSc courses.

[4]http://www.cs.man.ac.uk/peve/peveWebNew/home/index.htm
[5]http://www.cs.man.ac.uk/peve/peveWebNew/e_learning/index.htm
[6]http://www.cs.man.ac.uk/peve/peveWebNew/education/index.htm