Syllabus of MSc Course Units

2006/2007

The School of Computer Science

The University of Manchester

BMAN60112: IT Systems and Strategy

Level:	MSc
Credit Rating:	15 credits (7.5 ECTS)
Degrees:	Advanced MSc
Pre-requisites:	None
Teaching period:	1 days per week (5 weeks)
Coursework and exercises:	10 days
Assessment:	Exam 60%, Group project 40%
Lecturer:	Mr. Martin Cahill, Manchester Business School
Start time:	Starts at 1PM on the Monday.
Limit on numbers:	50 Participants

Introduction

A mixture of classroom lectures, video presentations, case studies, e-case studies, seminars, collaborative computing and on-line learning will be used to encourage a participative, discursive approach to this wide range of topics. e.g. Classroom projects, discussions and video.

Aims

It is now widely recognised that information is the lifeblood of companies. The focus to date has been on automating transactional based systems in all the business areas of a company such as production and logistics. The challenge for managers over the next decade is to build intelligence into their organisations that combine the best elements of integrated transaction based systems such as ERP, and banking systems, with knowledge based systems that support individual and group decision making, and enable the communication, storage and leverage of ideas and concepts across global enterprises.

The aim is to develop an understanding of key information systems strategy concepts, and contemporary developments in knowledge management.

Learning Outcomes

On completing the course unit students will have a critical understanding of:

- Key strategic concepts including changes in the business environment and company strategy in practice.
- Information Technology and competitive advantage, strategic alliances, planned versus emergent strategies, the relationships between business and information systems strategies.
- Electronic markets, electronic hierarchies and emerging network structures

- Project management applied to large-scale IT projects.
- The relationship between data, information and knowledge and the process of supporting decisions.
- Knowledge Management Systems and collaborative working tools.
- Approaches to electronic learning and electronic training

Course Content

Students should note that the syllabus is subject to modification.

Week 1: Strategy Concepts
Week 2: Information Systems Strategy
Week 3: Electronic Commerce
Week 4: Project Management and Implementation
Week 5: Fundamentals of Databases and Knowledge Management
Week 6: Data Warehouses and Data Mining
Week 7: Knowledge Management
Week 8: E-Learning and E-Training Systems

Reading List

Set text:
Pearlson, K. E. and Saunders, C. S. (2004) Managing and Using Information Systems: A Strategic Approach, John Wiley & Sons, Inc.

Recommended text:
Davenport, T. H. (1998) Working Knowledge: How Organizations manage what they know, Harvard Business School Press Boston, Massachusetts.

BMAN61051: IT Trends

Level:	MSc
Credit Rating:	15 credits (7.5 ECTS)
Degrees:	Advanced MSc
Pre-requisites:	None
Lecturer:	Dr. Nadia Papamichail, Manchester Business School

Aims

The course will address the role and importance of Information Systems in support of business, organisation and market strategies. It will provide an overview of current Information Technology (IT) components and trends and complement this with discussion and analysis of contemporary business developments including enterprise computing, global systems, electronic marketplaces and virtual organisations. It will also provide students with an awareness of the Internet and the World-Wide-Web as an information resource for business and academic research.

Learning Outcomes

On successful completion of the module students will have an understanding of the underlying technology and be able to assess its role and potential in business strategy and for achieving competitive advantage. Students will:

- have an understanding of business and information technology trends including an IT vocabulary
- have an appreciation of key technologies and their application to management problems
- understand the concept of strategic alignment and implementation issues

Course Content:

<u>Week 1, Information technology and business trends - 30 September</u>

Moore’s Law and its implications for price/performance ratios in business computing. Information technology costs as a percentage of firm revenue across different industries. The productivity paradox in the US and Europe. The IT industry, key players, contemporary changes and trends. Introduction to computer networks. The IT director/Chief information officer view of IT strategy.

Business 2015 project

In groups of 4-5 people, you should brainstorm the likely technology and business developments in 2015 for a specific industry. Groups will be allocated an industry from: manufacturing; retail; financial services, pharmaceuticals/health; and education. Each group will present the results of their findings to the rest of the class in a short 10 minute presentation. You should consider how IT and business are related to each other in a recursive manner (e.g. technologically determined business change, and business-led strategies that create new demands on the technology), and describe possible outcomes in the form of a scenario for a particular industry. The focus is on blue-sky thinking and the development of ideas to increase your understanding of the potential of IT in a business context.

Productivity paradox video, Money programme, September 1999

Reading:

1. www.businessweek.com/it100 this set of articles gives an overview of the IT industry in terms of product-markets, growth and size. You should try and understand the main market segments within the IT industry and gain an appreciation of the position of the top 10 vendors.

2. Building IT infrastructure for strategic agility, Peter Weill, Mani Subramani, Marianne Broadbent. *MIT Sloan Management Review*. Cambridge: Fall 2002. Vol. 44, Iss. 1; p. 57.
These authors have focused their research on IT infrastructure and its importance in business process design, business strategy deployment and strategic agility. It gives an overview of how companies view their IT in terms of categories of investment which is an important starting point for understanding IT costs, IT investment evaluation and strategic impact.

Week 2, Does IT matter? - 7th October

The strategic importance of IT for companies has always been vigorously debated by business leaders, consultants and academics. The dot.com boom-bust-boom cycle has arguably added confusion by adding conflicting evidence about the scale and growth of IT investment and the associated business outcomes. The ideas of Carr have given a new focus to the debate through his controversial assertion that IT has essentially become another commodity input to a business and that it will therefore not lead to competitive advantage. This will be used as the starting point for exploring the competitive impacts of IT that will be illustrated with a range of business examples and statistical data. Please read the CISCO case study in the context of Carr's thesis.

CISCO systems architecture: ERP and web-enabled IT, Ref. 9-301-099.
1. CISCO case study questions. Your group should prepare for a class presentation. You may be asked to answer any of the following questions.

(a) How are CISCO's business strategy and IT strategy related to each other?

(b) Map CISCO’s systems onto Keens’ Reach, Range, Responsiveness cube.
(c) Is IT a commodity resource for CISCO as outlined in Carr’s article?
(d) Apply the McKinsey 7S model to CISCO. How do you rank the relative importance of the elements?
(e) What are the HR implications of CISCO’s total reliance on networked information systems to manage every single business process?
(f) Discuss how CISCO could further develop its supply chain strategy.

Reading:

1. IT Doesn't Matter. By: Carr, Nicholas G.. *Harvard Business Review*, May2003, Vol. 81 Issue 5, p41, 9p.
This is the article that upset the IT leaders by claiming that IT is a simple commodity and therefore not of any great significant strategic benefit to companies. You should prepare a critical analysis of his argument, and identify any omissions.

2. The Engine That Drives Success, Source www.cio.com
The best companies have the best business models because they have the best IT strategies.
BY DON TAPSCOTT, May 2004.
This is probably one of the most convincing responses to Carr’s argument based on empirical examples.

3. Information technology and economic performance: A critical review of the empirical evidence, Jason Dedrick, Vijay Gurbaxani, Kenneth L Kraemer. *ACM Computing Surveys*. Baltimore: Mar 2003. Vol. 35, Iss. 1; p. 1
An academic study that gives an overview and synthesis of previous studies that provides some interesting conclusions based on secondary data.

4. Defining and measuring information productivity, Paul A. Strassman, 2004.
Paul Strassman has been one of the most influential writers on IT and its use by individuals, business organisations and Governments. In this paper he presents his own perspective and insights into the debate from the perspective of the CIO.

Week 3, Computing History and Internet Trends - 14th October

An overview of the history of business computing leading up to the emergence of the PC and the internet will be given in order to place the current IT industry and business developments in a broader context. The commercial potential of computers was recognised by only a very small number of pioneers and dismissed or massively underestimated by most business people. A common pattern that emerges from looking at historical comments of IT industry leaders up until the 1980s is that the level of understanding of how the future would evolve was extremely poor in nearly all cases. This begs the question of how much do we actually know and understand of what is likely to happen over the next ten to twenty years.

Video: Winners and losers in Internet Technology Companies

Computer network project: Smart business networks and networks of everything. How does the concept of a ubiquitous network that connects people, computers, products and everyday items affect the co-ordination of economic activity? Identify at least one new marketing opportunity for 2015 in retailing, manufacturing, pharmaceuticals, automotive, security and healthcare.

Reading:

1. Realising the Full Potential of the Web - Tim Berners-Lee, Director of the World-Wide Web Consortium (W3C)

Abstract
The first phase of the Web is human communication though shared knowledge. We have a lot of work to do before we have an intuitive space in which we can put down our thoughts and build our understanding of what we want to do and how and why we will do it. The second side to the Web, yet to emerge, is that of machine-understandable information. As this happens, the day-to-day mechanisms of trade and bureaucracy will be handled by agents, leaving humans to provide the inspiration and the intuition. This will come about though the implementation of a series of projects addressing data formats and languages for the Web, and digital signatures.
http://www.w3.org/1998/02/Potential.html

2. Platform 2015 Update: Technologies That Are "Aware" - Justin R. Rattner, Intel senior fellow and director of the Corporate Technology Group

Abstract
"We want to be able to design platforms that can anticipate and respond to the ever changing needs of the users, whether they're happening on the scale of minutes or hours, or at the fine grain, in terms of nanoseconds and microseconds. "
Source: http://www.intel.com/technology/techresearch/idf/fall-2005-keynote.htm

Week 4, Enterprise systems - 21st October

ERP systems dominate the corporate information technology landscape and represent the core systems for the majority of businesses. Compared with the early commercial system of BP, they represent a step change in the ambition of the IT industry and its customers to develop a globally comprehensive solution to an organisation's information systems requirements. The evolution of ERP systems will be analyzed from the perspective of the overall market, and the move towards packaged software, and from the perspective of individual companies operating in supply chains. A case study will be used to illustrate the implementation process of an ERP project, and its associated organisational and strategic impacts.

Reading:

1. ERP, Silicon Valley on the Rhine, Business Week. This is an introduction to the central idea of an ERP. The diagram gives an excellent overview of the key business processes in a typical ERP implementation.

2. European Threads and case vignette of Eurodiscount's implementation of a new IT strategy based on a best of breed strategy to replace ageing legacy systems. You should prepare the threads case for discussion in class.

3. Holland C.P. and B. Light (1999), "A Critical Success Factors Model for Enterprise Resource Planning (ERP) Implementation", IEEE Software, May-June, pp. 30-35.

Week 5, Systems design I, Dr. Peter Kawalek - 28th October

The basic systems design problem starts with the communication of concepts and ideas between general managers/non-technical staff and technical specialists and this is illustrated using a simple problem. The role of business process modelling and its possible use in overcoming the business-IT communication gap is discussed in the context of socio-technical thinking, and an overview of systems development methodologies is presented. Contemporary issues facing IT directors that are directly related to the design and evolution of information systems are outlined. These include: the growth in legacy systems; choice of broader systems design strategies; the growth of software packages and internet-based communication standards; and web services. To illustrate the organisational change aspects of systems design, a case study on one of the earliest large-scale commercial implementations will be used. BP Chemical's commercial information systems project illustrates the inter-relationships between strategy, organisational change and IT implementation.

Reading:

1. Laudon and Laudon, Ch. 3, Information systems and organisations, Ch. 8, managing data resources. This is an excellent introduction to how data/information is related to organisation design and operation.

2. Daft R.L. Organization Theory and Design, Ch. 7, "IT and knowledge management". On short loan collection, MBS library.
The chapter examines the impact of IT and knowledge management systems on organisations from an organisation theory perspective.

3. Davenport T.H. and J.E. Short "The New Industrial Engineering: Information Technology and Business Process Redesign", *Sloan Management Review*, Vol. 31(4), pp. 11-28.
This is the seminal article on how IT can be used to design organisations using business processes as the building blocks. This has arguably only

become possible with the advent of sophisticated enterprise systems which will be looked at in much more depth in week 4.

4. ERP Implementation Case Study. Kawalek, P., Wood-Harper, A.T., (2001) The Finding of Thorns: User Participation in an Enterprise Systems project, DATA BASE OF ADVANCES IN INFORMATION SYSTEMS, 33 (1)
'Rosebud' (pseudonym) is a hi-tech manufacturer which at one time carried out the largest ERP rollout in the world. The case study gives the inside story of how this was done, focusing on the way in which the implementation team engaged with the staff and managers in each new site.

Week 6, Systems Design II, Dr. Peter Kawalek, Multi-Media case group presentations - 4th November

Project Proof: Internet Enabled Process Reengineering at J.D. Edwards & Company, Volume 13 Article 30 May, 2004, **Nikunj Dalal** . This is a multi-media case study of the implementation of an ERP solution within a software company.

Project Proof Discussion Questions

1. How are the information systems and business strategies aligned and how is this measured?

2. How are business-processes designed and implemented?

3. What is the significance of vanilla business processes?

4. How do you rate the overall implementation strategy?

5. Why is implementation of ERP systems difficult and expensive?

6. What are the key drivers of cost and time?

7. What are the alternatives to ERP strategies?

Week 7, Strategy frameworks - 11th November

Dr. Jon Taylor, Co-operative Financial Services

Dr. Taylor will share his experiences about leading a major transformation project in financial services. He will outline the major aspects of the change programme which include information technology deployment and building new capabilities in areas such as HR, business processes and systems design.

Strategy

The central problem in information systems strategy is how to relate IT investments and strategies to business strategy. Several different approaches to this problem will be discussed including the McKinsey 7S model, MIT 90s framework, the socio-technical approach through to more recent developments such as Keen's Reach/Range/Responsiveness model and the business operating system. All of these approaches have a common theme of alignment. As the costs of IT investment have risen, a related problem of how to align the IT strategy with the business strategy is that of return on investment and how to measure the benefits of IT systems. There have been several approaches to the alignment problem ranging from the application of pure strategy models to develop IT strategies through to technology specific models such as ERP implementation models. An overview of the different approaches will be given and applied to companies in different sectors to illustrate the concepts. The tutorial reading by Markus (2000) gives an authoritative overview of how business and technology innovations are closely related to each other, and also proposes why problems still persist in IT deployment.

Reading

1. M. Lynne Markus (2000), Paradigm Shifts - E-Business and Business/Systems Integration**,** Volume 4 Article 10, Communications of AIS. http://cais.isworld.org/articles/4-10/default.asp?View=pdf&x=47&y=10

2. Peter G.W. Keen "Networks in Action", Chapter 3, Figure 3.7 (Reach, Range, Responsiveness).

Week 8, IT and competitive advantage - 18th November

The sources of change in global markets will be analysed and the particular role of information technology explored. The seminal articles by Porter on competitive strategy and information will be used to structure the session. Two important strategic implications of significant change in markets are instability and the concept of diverging returns. The sources of change will be discussed in the context of strategy examples from a range of different industries that illustrate the concepts of instability, diverging returns and average performance. The relationship between size and strategy will be discussed using ideas from economics, internet structure and the strategy literature. The concept of virtual size in a B2B context will be covered with examples from financial services and manufacturing. Notions of size for internet businesses will also be explored with evidence from retail internet marketing.

A business leader's perspective on IT trends and competitive advantage,
a presentation by Karl Wills, CEO of Abacus Billing

Karl Wills – Chief Executive Officer (CEO)

Karl Wills joined Abacus Billing in December 2000 and is responsible for managing the day-to-day operations of the company and overseeing the development of Abacus and the company.

Prior to joining Abacus Billing Karl had some 20 years experience in Senior Management roles for blue chip companies and a number of start-ups, with vast experience in pan-European and International management of technology, complex projects and logistics. Karl has held several high profile roles including Senior Logistics Strategy and IT Director at EMI International and as European Solutions Director at Metapack.

Karl has lived and worked in the Netherlands, France and Spain and has also run projects in most of Europe. He speaks reasonable Dutch and French. He is a graduate of Manchester Business School and has collaborated with various research projects in the IS group at MBS.

Reading

1. The Only Sustainable Edge: Why Business Strategy Depends on Productive Friction and Dynamic Specialization, Harvard Business School Press, John Hagel III, John Seely Brown (2005).

2. Holland C.P. and J.B. Westwood (2001), "Product-market and technology strategies in banking", *Communications of the ACM*, Vol. 44(6), pp. 53-57.

3. Porter M.E. and V.E. Millar (1985), "How information gives you competitive advantage", *Harvard Business Review*, Vol. 63(4), pp. 149-160.

Week 9, thebigword - 25th November

Thebigword is a global translation services company that has been at the forefront of exploiting IT for competitive advantage, particularly in terms of marketing. The company will be used to bring together the different technology and business strategy concepts based on the case study. There will also be a video of interviews with the senior management of the company to explain how the strategy frameworks have been applied in practice and how the high-technology marketing concepts are sold to global corporate clients.

Reading

Holland C.P., D.R. Shaw, J.B. Westwood and I. Harris (2004), "Marketing translations services internationally: exploiting IT to achieve a smart network", in Vervest et al (Eds.), Smart Business Networks, Springer Berlin, Heidelberg and New York. In short loan collection, MBS library.

E-case briefing.

Each group should prepare a strategic analysis of their allocated e-case company (e.g. Motorola). The specific brief is to explore the inter-relationships between the business strategy and the IT strategy of the firm, and discuss how the implementation of the overall strategy is managed. See http://www.mbs.ac.uk/webspace/pdrinkwater/e-Cases/e-cases.html for details of all of the e-cases. You must attend ALL of the presentations. A mark will be allocated to each group presentation and will form 40% of your overall mark, the remainder will be a written examination.

Week 10, E-cases presentations - 2nd December

Pedagogical Method

1) **Course Assessment:**

There are two assessments: group presentations in week 10 (40%), and an individual essay (60%).

2) **Learning Resource Details:** The reading for each lecture is given in the course outline above.

BMAN61102: Decision Analysis and Decision Support Systems

Level:	MSc
Credit Rating:	15 credits (7.5 ECTS)
Degrees:	ACSwICTM, ACS (and others with permission)
Pre-requisites:	None
Teaching Period:	1 day per week (5 weeks)
Coursework and exercises:	10 days (software development, case study, academic study, etc.)
Assessment:	33% exam, 17% group presentation, 50% project
Lecturer(s):	Prof. Simon French (Manchester Business School)
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 discuss 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 group presentation
learning outcomes (1), (2), (3), (4) and (5) by project

Contribution to programme learning

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

Reading list and supporting material

- P.R. Kleindorfer, H.C. Kunreuther, P.J.H. Schoemaker 'Decision Sciences: an integration perspective' Cambridge University Press 1993
- G.M. Marakas, Decision Support Systems in the 21st Century, Prentice Hall, 1999.
- E. Turban and J.E. Aronson (2001) Decision Support Systems and Intelligent Systems. 6th Edition. Prentice Hall.
- An extensive 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. Relationship with knowledge management systems

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. Deliberative e-democracy and e-participation

In depth case study of DSS for emergency management.

COMP60022: Grid Computing and eScience

Level:	MSc
Credit Rating:	15 credits (7.5 ECTS)
Degrees:	ACS/CompSciEng (possibly others)
Pre-requisites:	None
Teaching period:	1 day per week (5 weeks)
Coursework and exercises:	10 days
Assessment:	35% exam, 30% coursework 35% MiniProject
Lecturers:	Dr.John Brooke, 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 COMP 60202 web page. 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.

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. 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

COMP60031: High Performance Computing in Science and Engineering

Level:	MSc
Credit Rating:	15 credits (7.5 ECTS)
Degrees:	ACS, Computational Science and Engineering (and possibly others)
Pre-requisites:	None
Teaching Period:	1 day per week (5 weeks)
Coursework and exercises:	10 days
Assessment:	35% exam, 65% laboratory work and exercises
Lecturer(s):	Dr T L Freeman, Prof J R Gurd
Limit on numbers:	50 participants

Introduction

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

Aims

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

Learning Outcomes

A student completing this course unit should:

1. have an understanding of the different levels of abstraction in HPC Modelling and of different parallel programming models; (A)

2. have an understanding of parallel performance overheads and of techniques for reducing them; (A)

3. be able to implement a moderately complicated application in a parallel language; (B and C)

4. have an understanding of some parallel numerical algorithms; (A)

5. be able to work effectively as a member of a group to develop parallel applications. (D)

Assessment of learning outcomes

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

Contribution to programme learning

A1, A2, B2, B3, C1 and C3

Reading list and supporting material

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

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

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

Special resources needed to complete the course unit

Access to parallel computers situated in the School.

Detailed syllabus

Introduction to HPC

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

Levels of Abstraction, Models of Computation and Parallel Overheads

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

Restructuring for Parallel Performance

Parallelising Compilers; Loop Transformations; Data Transformations; Dependence Analysis; Compiler Strategies.

Parallel Algorithms

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

Resumé

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

COMP60042: Low-Power System Design

Level:	MSc
Credit Rating:	15 credits (7.5 ECTS)
Degrees:	ACS/CompSciEng (possibly others)
Pre-requisites:	None
Course material:	80 hours: on-line self-study material supported by unsupervised practical exercises and seminar sessions
Teaching period	1 day per week (5 weeks)
Coursework and exercises:	10 days
Assessment:	33% exam, 67% coursework
Lecturer(s):	Steve Furber
Limit on numbers:	50 participants

Introduction

This course covers the design of low-power embedded systems based around the ARM 32-bit microprocessor core. It will be taught primarily through self-study on-line material, supported by seminars and practical exercises.

Aims

Computing is becoming increasingly mobile, both in recognisable forms such as lap-top computers and in forms where the computing function is concealed such as digital mobile telephones. Mobile computing increases significantly the importance of minimising the power consumed by the system as excessive consumption directly compromises battery life. The aim of this course is to introduce students to the practical aspects of engineering high-performance computer systems where power consumption is a major consideration at every stage of the design. The course is heavily based around the ARM 32-bit RISC microprocessor, a world-leading processor for power-sensitive applications, and covers many aspects of designing power-efficient systems around ARM cores.

Learning Outcomes

A student completing this course unit should have achieved:

1. an understanding of the principles of the ARM and Thumb instruction sets and their practical use. (A)

2. an understanding of the principles of low-power RISC processor design. (A)

3. an insight into the design of memory hierarchies for power-efficient systems, and an ability to apply a systematic methodology to memory hierarchy design. (A and C)

4. an overview of the system-level issues involved in designing a particular power-sensitive application. (B)

5. an ability to write clear and concise reports on matters relating to low-power design. (D)

Assessment of learning outcomes

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

Contribution to programme learning

A1, A2, B2, C1, D3

Reading list and supporting material

1. Access to the course book is highly desirable: Furber, ARM System-on-Chip Architecture. Addison-Wesley, 2000.
2. Full course material, including the laboratory manual, are supplied on-line.

Special resources needed to complete the module

The ARM CBT (computer-based training package) contains the basic course material - this is available on-line.

Access to the ARM Developer Suite (ADS) is required for the practicals and the post-course work. This runs on Windows PCs and is available on School machines. A (limited-time) demo version may be available for part-time students wishing to take the course. Alternatively we can provide remote access to suitable School machines.

Detailed syllabus

Basics of processor design.

Processor design trade-offs.

The ARM and Thumb instruction sets in outline.

The ARM instruction set in detail.

Exceptions and special instructions.

The Thumb instruction set in detail.

ARM integer cores.

Memory hierarchy.

The ARM memory management and memory protection units.

ARM CPUs.

System development.

On-chip buses.

On-chip debug.

COMP60051: Visualisation for HPC

Level:	MSc
Credit Rating:	15 credits (7.5 ECTS)
Degrees:	ACS/CompSciEng (and possibly others)
Pre-requisites:	None
Teaching Period:	1 day per week (5 weeks)
Coursework and exercises:	10 days
Assessment:	33% exam, 67% practical exercise
Lecturer:	Terry Hewitt and Dr. N.W. John
Limit on numbers:	50 participants

Aims

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

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

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

Learning Outcomes

A student completing this course unit should:

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

Assessment of learning outcomes

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

Contribution to programme learning

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

Reading list and supporting material

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

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

Special resources needed to complete the course unit

The AVS visualisation software is used on this course unit.

Detailed syllabus

Lectures

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

Laboratory Sessions

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

Mini Projects

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

COMP60062: System Level Design

Level:	MSc
Credit rating:	15 credits (7.5 ECTS)
Degrees:	ACS/EEE (and others, if qualified)
Pre-requisites:	None
Teaching period:	5 days, 1 day/week: 20 hours lectures, 10 hours practical
Assessment:	33% exam, 67% coursework (laboratory reports)
Lecturer:	Prof. Steve Furber, Dr. Doug Edwards, Dr. Linda Brackenbury & Dr Jim Garside
Limit on numbers:	40 participants

Introduction

The design of a modern System-on-Chip (SoC) is a complex task involving a range of skills and a deep understanding of a hierarchy of perspectives on design, from processor architecture down to signal integrity. This course will provide insights into these processes, focussing primarily on the high-level issues of system modelling, IP core reuse, architecture modelling and testing, on-chip interconnect, and RTL synthesis.

Aims

This course unit aims to introduce the main tools and techniques employed at the
higher levels of complex SoC design, to provide an overview of the design process.

Learning Outcomes

A student completing this course unit should:

1. have knowledge and understanding of the principle tools used in system-level design

2. understand issues relating to on-chip interconnect, architecture modelling and testing, and design verification

3. understand the role of RTL synthesis, technology mapping, cell libraries and timing closure in the SoC design process

4. be able to apply this understanding to the design of prototype systems

5. have insights into future developments in SoC technology

Contribution to programme learning

A1, A2, B2, B3, C1, D1, D3

Assessment of learning outcomes

Learning outcomes A1, A2, B2, B3 are assessed by examination,
learning outcomes A1, A2, B2, B3, D1 are assessed by team presentations and learning outcomes A1, A2, B2, B3, D3 are assessed by laboratory reports.

Reading list and supporting material

Special resources needed to complete the course unit

Various specialist tools are used in the course laboratory: System C,
Verilog, ...

Detailed syllabus

What do you want? [4]
Specifications, system modelling with System C, IP blocks,
microprocessor
cores, on-chip interconnect (buses and Networks-on-Chip).

How do you design it? [4]
Architecture modelling and testing, tools and flows, verification.

Implementation. [4]
RTL synthesis, Verilog, design-for-test, low-power-design,
technology mapping, cell libraries, principles of tools, FPGAs.

Getting it to work. [4]
Debugging, timing closure, asynchronous design & GALS.

Where is it all going? [2]
The future of SoC design: reconfigurability, chip multiprocessors.

Feedback. [2]
Team presentations, working system demos.

COMP60071: Introduction to Computational Science

Level:	MSc
Credit Rating:	15 credits (7.5 ECTS)
Degrees:	Computational Science and Engineering (ACS and others, if qualified)
Pre-requisites:	None
Teaching period:	1 day per week (5 weeks)
Coursework and exercises:	10 days
Assessment:	35% exam, 65% laboratory work and exercises
Lecturer(s):	Dr T L Freeman, Professor J R Gurd
Limit on numbers:	50 participants

Aims

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

Learning Outcomes

A student completing this course unit should:

1. have an understanding of the execution cycle of a numerical code that simulates a simple ODE or PDE;

2. have an understanding of the factors in the execution cycle that affect performance on a sequential machine;

3. have a basic understanding of the fundamentals of computer architecture;

4. have an understanding of the benefits of abstraction in program design.

Assessment of learning outcomes

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

Contribution to programme learning

A1, A2, B2, C1 and C3.

Reading list and supporting material

M.T.Heath, Scientific Computing: An Introductory Survey, 2nd edition, McGraw-Hill, New York, 2002. ISBN 0-07-239910-4

D.Besset, Object-Oriented Implementation of Selected Numerical Methods: An Introduction with Java and Smalltalk, Morgan Kaufmann Publishers, 2000. ISBN 1558606793

D.Flanagan, Java in a Nutshell: a Desktop Quick Reference, 4th edition, O'Reilly, 2002. ISBN 0-596-00283-1

Detailed syllabus

Simple examples

Simple motivating examples (with simple boundary conditions):

- initial value ODEs,
- elliptic PDE (Laplace),
- parabolic PDE (heat conduction).

Discrete formulations

Step-by-step methods for ODEs. Finite difference approximations for PDEs; Linear equation solution methods (direct vs. iterative). Scaling with problem size (dependence of error on mesh size).

Difference Equations

Difference equations into code. Difference equations, systems of
linear algebraic equations, linear equation solution techniques (both
direct and iterative), Java code.

Details of solvers

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

Computer Architecture

Basic models of computation; basic von Neumann model; memory structure; system software.

Resumé

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

COMP60081: Fundamentals of High Performance Execution

Level:	MSc
Credit Rating:	15 credits (7.5 ECTS)
Degrees:	Computational Science and Engineering (ACS and others, if qualified)
Pre-requisites:	COMP 60071
Teaching period:	1 day per week (5 weeks)
Coursework and exercises:	10 days
Assessment:	35% exam, 65% laboratory work and exercises
Lecturer(s):	Professor J R Gurd, Dr T L Freeman,
Limit on numbers:	50 participants

Aims

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

Learning Outcomes

A student completing this course unit should:

1. have an understanding of the essentials of serial program execution cycle:

   source code ↳ object code ↳ run-time code ↳ hardware; (A)

2. have an understanding of the effects of components of execution cycle on performance;

3. have an understanding of the limitations of abstraction, in particular in terms of program performance. (D)

Assessment of learning outcomes

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

Contribution to programme learning

A1, A2, B2, C1 and C3.

Reading list and supporting material

Patt, Y.N. and Patel, S.J., Introduction to Computing Systems, McGraw-Hill, Boston, 2001.

Patterson, D.A. and Hennessy, J.L., Computer Organization & Design, 3rd Edition, Morgan Kaufmann, 2004.

Hennessy, J.L. and Patterson, D.A., Computer Architecture: A Quantitative Approach, 3rd Edition, Morgan Kaufmann, 2002.

Aho, A.V., Sethi, R. and Ullman, J.D., Compilers: Principles, Techniques and Tools, Addison Wesley, 1988.

Detailed syllabus

High level view of source code to run-time code transformation

Fundamentals of Compilation; Optimisation; Interpretation; Libraries.

High level view of run-time code to hardware transformation

Simple architectural model - one instruction at a time (issued at a constant rate), constant memory access time, performance implications.

Lower level view of source code to run-time code transformation

Formal description of program (Abstract Syntax Tree, Call Graph, Dependence Graph); Compiler Optimisations (Simple Data Dependences, Transformations, etc.); JIT Compilation.

Lower level view of run-time code to hardware transformation

More realistic (complex) architectural model - memory hierarchy (effects of cache), multi-way instruction issue, etc, performance implications. Evaluation of real performance effects on real hardware.

Resumé

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

COMP60092: Algorithms for Differential Equations

Level:	MSc
Credit Rating:	15 credits (7.5 ECTS)
Degrees:	Computational Science and Engineering (and others, if qualified)
Pre-requisites:	COMP 60071, COMP 60081
Teaching period:	1 day per week (5 weeks)
Coursework and exercises:	10 days
Assessment:	35% exam, 65% laboratory work and exercises
Lecturer(s):	Dr T L Freeman, Dr M Mihajlovic
Limit on numbers:	50 participants

Aims

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

Learning Outcomes

A student completing this course unit should:

1. have an understanding of the Numerical Analysis issues of algorithms for ODEs and PDEs (accuracy, stability, convergence);

2. have an understanding of the performance implications of algorithmic developments (algorithmic efficiency).

Assessment of learning outcomes

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

Contribution to programme learning

A1, A2, B2, C1 and C3.

Reading list and supporting material

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

Detailed syllabus

Initial Value Ordinary Differential Equations

Runge-Kutta methods and Multistep methods; accuracy, convergence and stability; error control; numerical examples.

Boundary Value Ordinary Differential Equations

Finite Difference and Finite Element Methods.

Elliptic Partial Differential Equations

Finite element methods; solution of large systems of algebraic equations; numerical examples.

Parabolic Partial Differential Equations

Finite element methods; analysis of error; numerical examples.

COMP60121: Automated Reasoning

Level:	MSc
Credit Rating:	15 credits (7.5 ECTS)
Degrees:	ACS (possibly others, if qualified)
Pre-requisites:	Propositional Logic (knowledge of predicate logic and some logic programming experience would be some advantage, but not essential).
Teaching period:	1 day per week (5 weeks)
Own work	1.5 days per week (week 1-5), 2.5 days per week in week 6.
Coursework and exercises:	10 days
Assessment:	40% exam, 60% laboratory work and exercises,
Lecturers:	Dr. Renate Schmidt and Dr. Alan Williams
Course unit webpage:	http://www.cs.man.ac.uk/~schmidt/COMP6012/
Limit on numbers:	50 participants

Introduction

Logic, the study of reasoning, plays an important role in theoretical computer science and many of the practical areas of computer science such as systems development, computer hardware, data bases, cognitive science, AI and logic programming. For example, in web technologies and agent technologies logical and automated reasoning methods are used for the intelligent processing of large ontologies, decision making based on knowledge bases of structured data, and formal specification and verification of multi-agent systems. An important part of the systems development process concerns reasoning about the behaviour of systems in order to verify the correctness of the behaviour. While these are specific examples of applications of automated reasoning, the focus in this course is on general aspects of logic and automated reasoning which are relevant to many such applications, as well as logic programming. The main motivation of the course is the study and development of general, efficient automated reasoning techniques.

Aims

Broadly, this course unit aims to provide an introduction to classical logic, automated theorem proving and logic programming. It aims to:

- provide students with an understanding of classical logic (propositional, first-order and clause logic),

- give an introduction to theoretical concepts and results that form the basis of current state-of-the-art theorem provers (and other theorem proving tools)
- discuss and study local reasoning methods (resolution) as well as global reasoning methods (tableaux)
- provide an introduction to logic programming.

It is assumed that students will be familiar with classical propositional logic (Boolean logic). Further knowledge of the subject is not assumed, although it would be an advantage for students to have some familiarity with predicate logic (first-order logic) and experience of logic programming, in particular Prolog (it is worth noting that students without such knowledge have not been disadvantaged in the past).

Learning Outcomes

A student completing this course unit should:

1. have knowledge and understanding of the syntax and semantics of classical propositional and predicate/first-order logic as well as clause logic (A).
2. have an understanding of the main ingredients of resolution calculi and be able to use them (transformation into clause form, inference rules, unification, orderings, selection) (A and B).
3. have an understanding of the main theoretical concepts for establishing refutational completeness of resolution calculi (candidate models, reduction of counter-examples) (A).
4. have an understanding of and be able to use the general concept of redundancy and be able to use it to justify different ways of simplifying and reducing the search space of theorem proving processes (tautology deletion, subsumption deletion, purity deletion, reduction) (A and B).
5. have an understanding of the main ingredients of semantic tableau calculi and establishing decidability and refutational completeness (inference rules, open/closed/strict/maximal tableau) (A).
6. be able to use the calculi covered in the course (resolution calculi, semantic tableau, free-variable tableau) for constructing proofs (B).
7. be able to use various systems (probably (M)SPASS and Vampire) and apply them to solve reasoning problems (C)
8. have an understanding of the relationship between resolution and logic programming (A).
9. have the competence to write Prolog programs (B and C).

Assessment of learning outcomes

Learning outcomes (1)-(6), (8), (9) will be assessed by exam. All learning outcomes will be assessed via paper and laboratory exercises set during the Teaching Period of the course unit.

Contribution to programme learning

A1, A2, B2, B3, and C3

Reading list and supporting material

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

- Kelly, J. (1997), *The Essence of Logic*. Prentice Hall.
- Schöning, U. (1989), *Logic for Computer Scientists*. Birkhäuser.
- Fitting, M. (1990), *First-Order Logic and Automated Theorem Proving.* Springer

The last two will appeal to students who find Kelly too basic.

A good introduction to logic programming and Prolog is available at: http://staff.science.uva.nl/~ulle/teaching/prolog/prlog.pdf

There are other text books available on the topics covered. Further details will be presented in the taught week.

Notes made available during the course unit to cover systems (probably (M)SPASS and Vampire) and all of the various topics presented in the course.

Detailed syllabus

The following lists the topics to be covered in the course. The Teaching Days will contain a mixture of lectures, examples classes and supervised laboratories. The numbers of sessions for each topic are given in brackets. If the topic is to be covered largely in the student's Own work time or in the labs, then this is also indicated.

The course unit is practicably-based, and so students will spend approximately one third of the total Teaching and Own Work Time undertaking laboratory exercises. This will include producing implementations of the various reasoning methods covered as well as using existing automated reasoning tools.

Introduction and motivation: [1].

Including example problems, problem representation via logic, computer assisted reasoning in mathematics.

Basics of sets and relations [Own].

What is a set, a relation, a function, set operations (intersection, union, etc), properties of binary relations (reflexivity, symmetry, transitivity, etc).

Revision of Propositional logic: [1].

Theory, language, models, validity and satisfiability, proof/inference rules, soundness and completeness, reasoning methods: truth tables, proof by contradiction, semantic tableaux.

Propositional logic reasoning using resolution: [1].

Normal forms, clauses, resolution.

First-order/predicate logic introduction: [1].

Quantifiers, first order models, validity and satisfiability.

First-order reasoning using unrestricted resolution [2].

Normal forms, clauses, Skolemization: elimination of quantifiers, unification, resolution, simplification techniques.

Orderings [1].

Well-founded orderings, multi-sets, multi-set orderings.

Refutational completeness of propositional resolution [3].

Herbrand interpretations, soundness, clause orderings, construction of candidate models, reduction of counter-examples, model existence theorem, refutational completeness, compactness of propositional logic.

Saturation-based framework of resolution calculi [2].

Ordered resolution with selection, lifting, refutational completeness, Craig interpolation, redundancy concept, saturation up to redundancy, practical model of a resolution prover, fairness, refinements of resolution, hyperresolution.

Formalisation of a concrete application [1].

Neuman-Stubblebine key exchange protocol.

Semantic tableaux [2].

Semantic tableau for propositional logic, decidability, refutational completeness, free-variable tableau, AMGU substitution rule, treatment of γ-formulae, refutational completeness.

Logic Programming: [2, supervised labs].

Horn clauses, SLD resolution, Prolog.

COMP60162: Knowledge Representation and Reasoning

Level:	MSc
Credit Rating:	15 credits (7.5 ECTS)
Degrees:	ACS and possibly others, if qualified.
Pre-requisites:	Some knowledge of logic and formal methods.
Teaching period:	1 day per week (5 weeks)
Coursework and exercises:	10 days
Assessment:	40% exam, 60% coursework
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:

have knowledge and understanding of the syntax and semantics of modal, description, and temporalised description logics, defaults, and formal concept analysis (A)

1. be able to formalise and represent knowledge in these logics and relate questions concerning this knowledge to logical reasoning problems (A and B)

2. have knowledge and understanding of a selection of logic-based applications (A and B)

3. 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)

4. 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 coursework exercises.

Contribution to programme learning

A1, A2, B2, B3, and C3

Reading list and supporting material

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

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

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

Modal Logic:

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

Description logics:

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

Further details will be presented in the taught week.

Detailed syllabus

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

Introduction and motivation.

Including example problems, problem representation via logic, computer assisted reasoning in mathematics.

Elementary set theory.

What is a set, a relation, a function, set operations (intersection, union, etc), properties of binary relations (reflexivity, symmetry, transitivity, etc).

Propositional logic.

Theory, language, models, validity and satisfiability, inference rules, soundness and completeness, reasoning methods: truth tables, proof by contradiction.

First-order logic.

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).

Additional Information

Additional information may be found at the course unit webpage.

COMP60171: Interactive System Design Methods

Level:	Advanced MSc
Credit Rating:	15 credits (7.5 ECTS)
Degrees:	ACS (possibly others, if qualified)
Pre-requisites:	Programme prerequisites only
Coursework and exercises:	10 days
Teaching period:	1 day per week (5 weeks)
Assessment:	100% coursework
Lecturer:	Dr Mark van Harmelen, mailto: markvanharmelen@yahoo.com

Introduction

Attend this module if you want to learn about

1. modern approaches to the design of the scope, content, functionality and user interfaces to interactive systems, and

2. the theory of interactive system design methods, particularly those in the field of object-oriented human-computer interaction (oohci) design.

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

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

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

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

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

Aims

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

Learning Outcomes

A student completing this course unit should:

1. Have knowledge and understanding of the principles of interactive system design, particularly those within the oochi design tradition. (A)

2. Understand how to perform, and have experience in performing, interactive system design using a participatory oohci design method. (B2,B3,C1,C3,C4,D)

3. Have at least a reading knowledge of the Unified Modeling Language, if not the ability to write analysis level system descriptions in UML. (A2)

4. Understand what constitutes an oohci method, and have the ability to design their own oohci method. (A,B2)

Assessment of learning outcomes

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

Contribution to programme learning

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

Reading list and supporting material

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

UML Slides

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

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

HCI design slides

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

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

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

Basics of HCI methods

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

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

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

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

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

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

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

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

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

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

Design techniques and methods

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

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

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

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

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

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

Muller, M.J. Retrospective on a year of Participatory Design using the PICTIVE Technique, Proc CHI'93, ACM, 1993.

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

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

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

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

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

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

Evaluation techniques

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

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

Some evaluation techniques.

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

Keen students find and read this.

Object modelling and human-computer interaction (oohci) design

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

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

Participatory practice

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

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

Special resources needed to complete the module

None.

Detailed syllabus

Introductory topics

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

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

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

Human-computer interaction and interaction design.

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

Participatory design

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

An example oohci method

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

A formal treatment of oohci methods

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

Introductory work

Read the introductory reading as printed and distributed by the School (collect from room 2.3).

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

This is a time breakdown of what you should do during 60 hours introductory work:

20 hrs:	Read, practice and learn object modelling using UML notes
28 hrs:	Read supplied reading pack (see above).
4 hrs:	Draw a rich picture that describes use of The Bridge. Then draw three rich pictures, one for each stage of The Bridge. Rich Pictures are described in the reading pack.
8 hrs:	Spread over the pre-course work time period: research the web, noting interesting resources for cs617.

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

This reading is the topic of the first continuous assessment test.

COMP60242: Mobile Computing

Level:	MSc
Credit Rating:	15 credits (7.5 ECTS)
Degrees:	ACS (and others, if qualified)
Pre-requisites:	Basic mathematics
Teaching period:	1 day per week (5 weeks)
Coursework and exercises:	10 days
Assessment:	67 % laboratory and 33 % exam
Lecturer(s):	Dr. Barry Cheetham
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

1. At the end of the course, students will have acquired the following knowledge and skills.
2. Understanding of characteristics of radio propagation and interference in multipath propagation and channel model description (A1,A2)

3. 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)

4. Understanding of the issues and techniques used in the design of Medium Access Control protocols for wireless Networks (A1,A2)

5. Understanding of the systems, protocols and mechanisms to support mobility for mobile internet users (A1,A2)

6. The ability to investigate fundamental aspects of transmission and modulation by writing MATLAB programs. The experience of using an industrial standard network simulation package, such as OPNET(B1,C1,C2,D4)

Assessment of learning outcomes

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

Contribution to programme learning

A1,A2, B1,C1,C2,D4

Reading list

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

Supplemental books

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

Detailed Syllabus

Introduction to wireless networking.

Advantages and disadvantages of wireless networking

Characteristics of radio propagation.

Fading, Multipath propagation

Introduction to digital transmission.

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

Narrowband digital modulation.

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

Wideband modulation techniques to cope with intersymbol interference

Direct sequence spread spectrum Adaptive Equalization Orthogonal frequency division multiplex

Medium Access Control (MAC).

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

Protocols supporting mobility.

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

COMP60312: Computational Biology - The application of computer science to the problems of post-genome biology

Level:	MSc
Credit Rating:	15 credits (7.5 ECTS)
Degrees:	ACS and others
Pre-requisites:	A knowledge of modern biology is not a course prerequisite.
Teaching period:	1 day per week (5 weeks)
Coursework and exercises:	10 days
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)

COMP60321: Computer Animation

Level:	MSc
Credit Rating:	15 credits (7.5 ECTS)
Degrees:	ACS/CMIM/CompSciEng and others
Pre-requisites:	Students taking this course must have some previous experience of computer graphics programming.
Teaching period:	1 day per week (5 weeks)
Coursework and exercises:	10 days
Assessment:	50% exam, 50% practical work
Lecturer(s):	George Leaver (MC) with Martin Turner (MC)
	Nigel John (University of Wales, Bangor)
Limit on numbers:	25 participants

Aims

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

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

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

Learning Outcomes

A student successfully completing this course unit will:

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

Assessment of learning outcomes

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

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

Reading list and supporting material

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

John Vince, Essential Computer Animation Fast;

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

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

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

Special resources needed to complete the course unit

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

Detailed syllabus

Lectures:

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

    - Pixar
    - Final Fantasy
    - Other key examples from recent productions.

Laboratory Sessions:

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

COMP60342: Electronic Commerce Technologies

Level:	MSc
Credit Rating:	15 credits (7.5 ECTS)
Degrees:	ACS and others
Pre-requisites:	Some knowledge of data modelling and database technologies helpful.
Teaching Period:	1 day per week (5 weeks)
Coursework and exercises:	10 days
Assessment:	30% exam, 20% lab, 50% coursework
Lecturer(s):	Dr RW Giordano (r.giordano@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 in “Web 2.0” environments; using database technologies to support web-based interactions and services; modelling data and processes; evaluating web-based products 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 integrate technologies and resources 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 designed, built and implemented. (A)

2. have a knowledge of tools, technologies, concepts and processes, that comprise the technical infrastructure of eCommerce sites and be able to solve problems about site design, hardware and software architecture, and document architecture. (A and B)

3. have a knowledge of data architecture and be able to solve problems about modelling data and processes so that they can be discovered in web-based environments (A and B)

4. be able to design an ecommerce or advanced web application and evaluate and justify the design. (B)

5. be able to encode data in XML and prepare a technical report on the modelling and ontologies as they relate to a web site in question. (C)

6. be able to work effectively as a member of a group to design and implement a web-based application in a real-world environment. (D)

Assessment of learning outcomes

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

Contribution to programme learning

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

Reading list and supporting material

Eve Andersson, Philip Greenspun and Andrew Grumet. *Software Engineering for Internet Applications* (Cambridge, MA; MIT Press, 2006) ISBN: 0262511916. This book is free on the web at <http://philip.greenspun.com/seia/>. Students are urged to become familiar with Ruby-On-Rails and to visit <http://www.rubyonrails.org>. The lecturer produces a 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. We plan to use Ruby-On-Rails as our development environment.

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.

Web 2.0, social software, electronic communties and markets [1].

Building, recognizing, managing and making use of online communities in web-based environments (such as communities of practice, communities of purpose, peer communities, knowledge communities, etc.) and their relationship to ecommerce.

Web architecture. [1].

Structural design of eCommerce systems. Client-server architecture, 2-, 3-,n-tier design, server farms, scalability. Integration of legacy systems. Particular problems posed by 24/7operation and an open user community.

Web Services [2].

Standards and processes that support data interchange, remote program invocation, self-description and universal discovery.

Usability [2].

User-interface design for web-sites with a focus on increasing productivity and conversion rates (from visitors to customers); mobile applications; applications for persons with special needs.

Evaluation Techniques [1].

Understanding the roles of product and process evaluation to increase quality; heuristic evaluation; establishing specification metrics; competitive benchmarking.

COMP60362: Advanced Database Technologies

Level:	MSc
Credit Rating:	15 credits (7.5 ECTS)
Degrees:	ACS/CMIM and others
Pre-requisites:	Good familiarity with relational databases and programming.
Teaching period:	1 day per week (5 weeks)
Coursework and exercises:	10 days
Assessment:	33 % exam, 67 % coursework
Lecturer(s):	Dr Ulrike Sattler and Dr Alvaro A. A. Fernandes
Course Unit webpage:	http://www.cs.man.ac.uk/~sattler/teaching/cs636.html
Limit on numbers:	50 participants

Introduction

This course unit is divided into two parts: one on Semi-Structured Data and one on Data Mining. Both semi-structured data and data mining are advanced recent developments in database technology which aim to address the problem of extracting information from the overwhelmingly large amounts of data which modern societies are capable of amassing. Semi-structured data focuses on describing and querying data that comes in a format less tightly structured than that found in relational databases. Data mining focuses on inducing compressed representations of data in the form of descriptive and predictive models.

Aims

The semi-structured data part of the course unit aims to give students a good overview of the ideas and the techniques which are behind the description and query mechanisms for semi-structured data. We discuss semi-structured data and their representation, XML, Schemata for XML data (DTD and XMLSchema), processing and manipulating XML data (XPath, XQuery), and some theoretical aspects of XML data processing. Laboratory sessions will ground the abstract notions on practical cases and tools.

The data mining part of the course unit aims to motivate, define and characterize data mining as a process; to motivate, define and characterize data mining applications; to survey, and present in some detail, a small range of representative data mining techniques and tools. Laboratory sessions will ground the abstract notions on practical cases and tools.

Learning Outcomes

A student completing this course unit should:

1. have an understanding of the foundations semi-structured data and their representation, XML, Schemata for XML data (DTD and XMLSchema), processing and manipulating XML data (XPath, XQuery), and some theoretical aspects of XML data processing. (A)

2. have mastered the basic range of techniques for representing, modelling, and querying semi-structured data, and be able to use tools developed for them. (B, C and D)

3. have an understanding of the data mining process, its motivation, applicability, advantages and pitfalls. (A)

4. have an understanding of the principles, methods, techniques, and tools that underpin successful data mining applications. (A and C)

5. be able to apply the methods and techniques surveyed in the course using a data mining workbench. (B, C and D)

Assessment of learning outcomes

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

Contribution to programme learning

A2, B2, B3, C2, D3, D4

Reading list and supporting material

For the SSD and XML part of the course unit:

- J. Simeon and P. Wadler. The Essence of XML. In POPL, 2003.
- S. Abiteboul, P. Buneman, D. Sucium. Data on the Web. Morgan Kauffman, 2000.
- more to follow

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

- I. Witten and E. Frank, Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufman, 1999. *This is the one that lectures notes are most closely based on.*
- J. Han and M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufman, 2000. *This is more database-centred, in contrast to Witten and Frank, who take a machine-learning viewpoint of data mining. It is also useful in covering data warehouses too, to some extent.*

- D. Hand, H. Mannila and P. Smyth. Principles of Data Mining, MIT Press, 2001. *This takes yet another viewpoint on data mining, viz., the statistical one. In this sense, it is the least related to the approach followed in this part of the course.*
- M. H. Dunham. Data Mining: Introductory and Advanced Topic. Prentice Hall, 2003. *This has yet another slight shift in emphasis, as it more or less favours an algorithmic viewpoint and is, in this sense, a core computer-science view of the issues.*

Special resources and limits on participation

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

Detailed syllabus

Part I

- Introduction [2]: Semistructured Data and XML.
- Schema languages [3]:
    - DTDs and validation of XML documents with respect to a DTD.
    - XML Schema.
- Query languages [5]:
    - XPATH
    - XQuery
    - XPath Containment
- Storing and querying XML in relational databases [1]

Part II

- Introducing Data Mining [1]: Why data mining?; What is data mining?; A View of the KDD Process; Problems and Techniques; Data Mining Applications; Prospects for the Technology.
- The CRISP-DM Methodology [1]: Approach; Objectives; Documents; Structure; Binding to Contexts; Phases, Task, Outputs.
- Data Mining Inputs and Outputs [3]: Concepts, Instances, Attributes; Kinds of Learning; Providing Examples; Kinds of Attributes; Preparing Inputs. Knowledge Representations; Decision Tables and Decision Trees; Classification Rules; Association Rules; Regression Trees and Model Trees; Instance-Level Representations.
- Data Mining Algorithms [3]: One-R; Naïve Bayes Classifier; Decision Trees; Decision Rules; Association Rules; Regression; K-Nearest Neighbour Classifiers.
- Evaluating Data Mining Results [3]: Issues in Evaluation; Training and Testing Principles; Error Measures, Holdout, Cross Validation; Comparing Algorithms; Taking Costs into Account; Trade-Offs in the Confusion Matrix.

Additional Information

Additional information may be found at the course unit webpage.

COMP60391: Computer Security

Level:	MSc
Credit Rating:	15 credits (7.5 ECTS)
Degrees:	All Advanced Masters
Pre-requisites:	None
Teaching period:	1 day per week (5 weeks)
Coursework and exercises:	10 days
Assessment:	30% in-class test, 70% coursework
Lecturers:	Daniel Dresner (National Computing Centre) daniel.dresner@ncc.co.uk; Ning Zhang (CS) and invited speakers.

Introduction

Note: This course unit is jointly developed and delivered by Ning Zhang (from the School of Computer Science) and Daniel Dresner (the Standards Manager at the National Computing Centre, an independent research organisation which promotes the effective use of information technology).

Aims

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

Learning Outcomes

A student successfully completing this course unit should:

1. have a good understanding of how to define system security requirements and a good understanding of a variety of generic security threats and vulnerabilities, and be able to identify and analyse particular security problems for a given application [A and B];

2. be able to prioritise requirements, and match requirements to solutions and countermeasures commensurate with associated risks [B,C,D];

3. have a good understanding of the correlation of business processes to technology in relation to security requirements [A];

4. be familiar with the relevant industry security standards and the regulation, and their application [C];

5. Appreciate the application of security techniques and technologies in solving real-life security problems in practical systems [A].

Assessment of learning outcomes

Learning outcomes (1), (2), (3), (4) and (5) will be assessed by in-class tests and the evaluation of a system security management plan for a case study.

Contribution to programme learning

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

Reading list and supporting material

The following material supports the course unit:

- National Computing Centre Guideline 275, *Desert Island Standards*, February 2003.
- National Computing Centre Guideline 269, Managing Risk - a practical guide, July 2002.
- ISO 17799: Code of practice for information security management.
- W. Stallings, Cryptography and Network Security, 4th/e, ISBN: 0-13-187316-4, Prentice Hall, 2006.
- Dresner, Daniel; Information Security Management, ISBN 0-85012-885-4, National Computing Centre, 2006
- Graff, Mark G., and van Wyk, Kenneth R., Secure Coding Principles and Practices, ISBN 0-596-00242-4, O'Reilly and Associates Inc, 2003

Detailed syllabus

1a. The need for information assurance

Security breaches. Introduction to business continuity. System lifecycles. Trust.

1b. Introduction to standards

Plan-do-check-act lifecycles. Overview of ISO 27001/ISO 17799 Information security management.

2. Information security management

Security policy. Security organisation. Asset management and control. Human resources security. Physical and environmental security. Communications and operations management. Access control. System acquisition, development and maintenance. Information Security Incident Management. Business Continuity management. Compliance.

3. Risk management

4a. Vulnerabilities

Windows, Unix and Open source.

4b. Solutions and countermeasures

Entity authentication. Message Security. Intrusion detection/prevention. Firewalls. Anti-virus software. Virtual Private Networks (VPNs).

5. Active security

Audits and reviews: Vulnerability scanners, Penetration testing. Inspection.

COMP60431: Machine Learning

Level:	MSc
Credit Rating:	15 credits (7.5 ECTS)
Degrees:	ACS/CompSciEng (and others, if qualified)
Pre-requisites:	None
Teaching period:	1 day per week (5 weeks)
Coursework and exercises:	10 days
Assessment:	33% exam, 67% coursework (laboratory reports)
Lecturer:	Dr. Magnus Rattray and Dr. Aphrodite Galata
Limit on numbers:	50 participants

Introduction

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

Aims

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

Learning Outcomes

A student completing this course unit should:

1. have knowledge and understanding of the principle algorithms used in machine learning, as outlined in the syllabus below (A)

2. have sufficient knowledge of information theory and probability theory to provide a theoretical framework for machine learning (A)

3. be able to apply machine learning algorithms, evaluate their performance and appreciate the practical issues involved in the study of real datasets (C)

4. be able to provide a clear and concise description of testing and benchmarking experiments (D)

Assessment of learning outcomes

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

Contribution to programme learning

A1, A2, C1, D3, D4

Reading list and supporting material

There is a CS643 web page with further details for the current session. The main course textbook is

Alpaydin, E., ``Introduction to Machine Learning'' MIT Press, 2004. This is the new course textbook and covers a very broad range of machine learning topics.

Additional reading

Mitchell, T. M., ``Machine Learning'' McGraw-Hill, 1997. Introduction to machine learning, covering a broad range of topics and algorithms. This was the previous course textbook and provides an accessible introduction to many of the key concepts.

Hastie, T., Tibshirani, R., Friedman, J., ``The Elements of Statistical Learning'' Springer, 2001. An advanced textbook taking a statistical perspective.

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

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

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

Special resources needed to complete the course unit

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

Detailed syllabus

Introduction to machine learning [1].

Overview of different tasks: classification, regression, clustering, control.

Concept learning, information theory and decision trees [2].

Concept learning (algorithms and limitations), Shannon's entropy, mutual information and gain, ID3 and extensions.

Introduction to probabilistic modelling [2].

Probability distributions and densities, Bayes' rule, maximum likelihood, Bayesian inference.

Unsupervised learning [2].

Clustering (Gaussian mixtures, EM-algorithm, k-means), Dimensionality reduction (PCA).

Non-linear regression and classification [2].

Feed-forward neural networks, support vector machines.

Sequence learning [2].

Markov chains, hidden Markov models.

COMP60442: Advanced Machine Vision

Level:	MSc
Credit Rating:	15 credits (7.5 ECTS)
Degrees:	ACS/CMIM and others, if qualified
Pre-requisites:	Mathematics, C or Matlab programming
Teaching period:	1 day per week (5 weeks)
Coursework and exercises:	10 days
Assessment:	40% exam, 30% lab, 30% coursework
Lecturer(s):	Dr. A. Galata (agalata@cs.man.ac.uk), with 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:

have an understanding of commmon machine vision algorithms. (A)

have a knowledge of the statistical design of algorithms. (A and B)

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)

be able to design basic systems for image analysis and evaluate and justify the design. (B)

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)

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.tina-vision.net/teaching/cvmsc/index.html

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.

COMP60461: The Semantic Web: Ontologies and OWL

Level:	MSc
Credit Rating:	15 credits (7.5 ECTS)
Degrees:	ACS and others, if qualified
Pre-requisites:	A knowledge of basic logic
Teaching period:	1 day per week (5 weeks)
Coursework and exercises:	10 days
Assessment:	40% exam, 30% Taught Week Labs and Exercises, 30% Post-course work
Lecturers:	Sean Bechhofer and Alan Rector.
Limit on numbers:	50 participants

Introduction

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

Aims

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

Learning Outcomes

A student successfully completing this course unit should:

1. Be able to discuss/explain the general principals of semantic networks, frames, rules (A),

2. Be able to discuss/explain KR/ontology languages designed for the world wide web, in particular the new Web Ontology Language (OWL) (A, B),
3. Understand the syntax, semantics and decision procedures for the famly of description logics which underpin OWL (A),

4. Know the common ontological structures and principles of ontology development , have an appreciation of ``why it's hard'', and to be able to write critically about current work on the ``Semantic Web'' (A, B),

5. Be able to design and build ontologies in OWL using the *de facto* standard editor, OilEd, justify and evaluate their design (B, C), and explain their behaviour.

Assessment of learning outcomes

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

Contribution to programme learning

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

Reading list and supporting material

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

Ian Pratt. Artificial Intelligence. Macmillan, 1994.

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

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

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

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

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

- Ian Horrocks and Peter F. Patel-Schneider. Three theses of representation in the semantic web. In *Proc. of the Twelfth International World Wide Web Conference (WWW 2003)*, 2003.

Detailed syllabus

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

Precourse Work

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

Background review of logic

Frames and Semantic Nets

Initial Protege tutorials

Course work

Basics of knowledge representation and informal introduction to OWL

Description logics and classiers - the ALC family and its extensions

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

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

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

Laboratory Work

Introduction to Protege and OWL including advanced tutorial

Special problems of representation and reasoning in OWL

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

Practical development project

Problem sets

Critique/comment on implemented ontologies on the Web

Postcourse work

Practical development project

Problem sets

Critique/comment on implemented ontologies on the Web

Additional information may be found here.

COMP60492: Robotics

Level:	MSc
Credit Rating:	15 credits (7.5 ECTS)
Degrees:	ACS (and others, if qualified)
Pre-requisites:	None
Teaching period:	1 day per week (5 weeks)
Coursework and exercises:	3 weeks
Course work:	Robot design and analysis
Assessment:	70% 2h exam, 10% Lab, 20% course work,
Lecturer:	Dr Robert Richardson
Limit on numbers:	20 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.

COMP60992: Research and Professional Skills

Level:	MSc
Credit Rating:	None
Degrees:	All degree programmes except MEnt
Pre-requisites:	None
Teaching period:	Spread over the academic year: Mixed activities - lectures, seminars, group skill activities
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.

COMP61001: Introduction to Advanced Computer Science

Level:	MSc
Credit Rating:	None
Degrees:	ACS/CS/CompSci
Pre-requisites:	None
Teaching Period:	1 day per week (5 weeks)
Assessment:	None
Lecturer(s):	Programme Director, Tutor and Lecturers.

Introduction

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

Aims

This course unit has three aims.

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

Learning Outcomes

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

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

Assessment of learning outcomes

There is no assessment for this course unit.

Reading list and supporting material

Lecturers will supply material about each of the course units.

Special resources needed to complete the course unit

No special resources.

Detailed syllabus

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

COMP70042: Low-Power System Design

Course Duration:	17-18 weeks
Required Time per Week:	8 - 10 hours.
Course Contact:	Stuart Anderson (stuart@cs.man.ac.uk)
Summary:	The aim of this course is to introduce students to the practical aspects of engineering high-performance computer systems where power consumption is a major consideration at every stage of the design. The course is heavily based around the ARM 32-bit RISC microprocessor, a world-leading processor for power-sensitive applications, and covers many aspects of designing power-efficient systems around ARM cores.
Prerequisites:	A general familiarity with microprocessor systems is assumed.

Objectives:

A student completing this course unit should have achieved:

- an understanding of the principles of the ARM and Thumb instruction sets and their practical use.
- an understanding of the principles of low-power RISC processor design.
- an insight into the design of memory hierarchies for power-efficient systems, and an ability to apply a systematic methodology to memory hierarchy design
- an overview of the system-level issues involved in designing a particular power-sensitive application (a GSM digital mobile phone handset).
- a preliminary view of the potential role of asynchronous design in future low-power systems.
- an ability to write clear and concise reports on matters relating to low-power design.

Course Content:

Computing is becoming increasingly mobile, both in recognisable forms such as lap-top computers and in forms where the computing function is concealed such as digital mobile telephones. Mobile computing increases significantly the importance of minimising the power consumed by the system as excessive consumption directly compromises battery life.

The full syllabus is as follows:

Topic	**Content**
ARM assembly language programming	ARM software development tools; the ARM programmers' model; the ARM instruction set; writing simple programs; example programs.
Support for high-level languages	ARM data types; memory organization; high-level language support.
The ARM instruction set in detail	Operating modes and exceptions; conditional execution; instruction types and functions.
The Thumb instruction set	The Thumb programmers' model; Thumb instructions; Thumb implementation; Thumb applications; example Thumb programs.
ARM integer cores	ARM organization; the ARM 3- and 5-stage pipelines; the ARM7TDMI core; the ARM9TDMI core.
Architectural extensions	The ARM coprocessor interface; floating-point support; DSP support.
Memory hierarchy	On-chip RAM; caches; memory management; operating systems; the ARM system control coprocessor; the ARM MMU architecture; the ARM memory protection unit.
ARM CPUs	The ARM700 series; the ARM810; StrongARM; the ARM900 series; the ARM1020.
System-on-Chip development	The ARMulator; JTAG test access port; embedded core debug; embedded trace; the AMBA on-chip bus; hardware prototyping; examples of embedded system chips.
Case study - a GSM handset	The GSM digital cellular network; handset organisation; hardware/software trade-offs; power minimisation.
Asynchronous design for low power	Motivation for asynchronous design; asynchronous design styles; micropipelines.
The AMULET microprocessors	AMULET1 organisation; AMULET1 characteristics; lessons from AMULET1; the AMULET2e asynchronous embedded controller; the AMULET3H SoC subsystem; the DRACO chip.

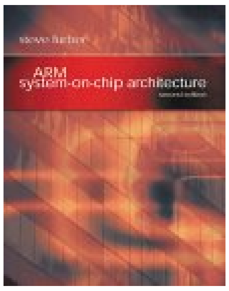
Assessment:

100% assessed excercises. Assignments (typical):

- Assignment 1 is concerned with the analysis of cache memory behaviour.
- Assignment 2 is concerned with the analysis and optimisation of a representative low power application program.

Further Details:

Recommended Texts:



- ARM System-on-Chip Architecture
- By Steve Furber
- Publisher: Addison Wesley
- ISBN: 0201675196

Full course notes, including the laboratory manual, will be supplied as work packages in Text and/or PDF format.

Delivery Mode:

- Fully on-line distance learning with on-line support.
- 10 hours of on-line tutorials.
- Bulletin boards on-line.
- Work package documents supplied on-line.
- Course material – course notes and CBT package supplied on-line.
- On-line examination.
- Course book – paper based.

Prices & Bookings Information

COMP70101: Programming in C

Course Duration:	16-17 weeks
Required Time per Week:	8 - 10 hours.
Course Contact:	Chris Page (cpage@cs.man.ac.uk)
Summary:	This course is designed to develop the skills required to enable participants to write programs using the C Programming Language. It will run entirely on the Web, using a Virtual Learning Environment and a specifically developed Computer Based Training Package.
Prerequisites:	The only formal prerequisite is basic computer literacy. However, the course is designed for people with a good general education (e.g. a degree or good 'A' levels). The ability to think logically and solve abstract problems is a prerequisite for computer programming in any language.

Objectives:

A student completing this course unit should:

- have an understanding of the main programming constructs of C.
- have an understanding of the role of design in the development of programming solutions to problems.
- have a knowledge of some standard algorithms and data structures in imperative programming and be able to solve problems using lists, trees and recursion.
- have the competence to write programs in C.

Course Outline:

The full syllabus is as follows:

Topic	**Content**
Introduction	What is C? Basics of program writing.
Information representation	Variables, data types, memory allocation, pointers, strings, arrays and structures.
Control flow	Expressions, control statements.
Program structuring	Functions headers, independent compilation and variable scope.
Input and Output (I/O)	Character streams, standard input/output and file input/output.
Design techniques	The role of design in the development of solutions to

and modularisation	problems. Top down structured design. Modular design including the use of separate compilation facilities for C.
Algorithms and dynamic data structures	An outline will be given of the role of standard algorithms and dynamic data structures (linked lists, trees, recursion).
Advanced I/O (optional material)	Input and output buffering and miscellaneous input and output functions.

Assessment:

40% programming exercises, 60% projects.

Further Details:

The taught half of the course is divided into 18 work packages. For each work package there are a number of modules of a Computer Based Training (CBT) package to complete, an example program to understand, compile and run and an exercise to carry out. After every four work packages the exercise will be assessed.

Within the course you will be guided to the appropriate modules in the CBT package to work through prior to looking at the provided example and completing the specified exercise. You will interact with the members of the course team, and with other learners, through email, a course bulletin board and 'chat rooms' for on-line tutorials.

Prices & Bookings Information

COMP70111: Introduction to software development in Java

Course Duration:	16-17 weeks
Required Time per Week:	8 - 10 hours.
Course Contact:	John Sargeant (johns@cs.man.ac.uk)
Summary:	To ensure that students have a thorough grasp of the basics of object-oriented programming in Java 1.5. The emphasis is on fundamental principles and their application in practice. Language constructs and library classes are introduced as embodiments or examples of the principles and best practice is emphasised throughout.
Prerequisites:	None.

Objectives:

A student successfully completing this module should be able to:

- Explain the relationship between real - world entities and software objects with suitable examples.
- Make appropriate use of existing classes
- Write simple classes to model application domain concepts.
- Use the basic imperative features of Java with confidence
- Create programs consisting of small collections of classes, which obey the basic best practice rules of responsibility assignment, low coupling etc.
- Build simple inheritance hierarchies which pass the is-a test
- Handle runtime errors using exception handling in accordance with best practice
- Create very simple GUI applets and applications using Swing
- Perform basic stream I/O and file handling
- Calculate the complexity of simple algorithms using collections and explain why it matters

Course Outline:

The full syllabus is as follows:

Topic	Content
Object-oriented basics	• What is Java? • Mental models - how we deal with the world • Software objects - mental models on a computer • Creating objects and sending messages • A complete simple classImportance of

	documentation - javadoc • Other ways of programming - and why OO is better! (*optional*)
Imperative programming	• Nuts and bolts (scalar values and expressions) • Handling text (Strings and the magic **+**) • Saying things and doing things (declarations and statements) • Making choices (if and switch) • Repeated computation (while and for) • The simplest collection (arrays) • How fast does it go? (A first look at complexity) • Dividing up the job (procedural abstraction, parameter passing)
Classes, responsibilities and collaborations	• Alternative implementations (encapsulation) • Alternative interfaces (overloading) • When are two objects the same (object references, equality vs. identity) • Assigning responsibilities to classes (which methods go where, unit testing) • Collaborating classes to solve problems (putting it all together, system testing) • What if there's no object to send a message to (static things) • Larger-scale organisation (packages)
Inheritance	• Mental models revisited - is-a-kind-of hierarchies • Abstract classes (representing common abstractions) • Extending classes (concrete sub-concepts) • The way objects understand messages (static checking, dynamic binding) • What have we inherited? (inheritance semantics) • When to use inheritance (is-a test, evils of implementation inheritance) • Interfaces (in the Java-specific sense)
Exception handling	• What if unexpected things happen at runtime? • Basic constructs - try.. catch.. finally, exception propagation • Throwing exceptions and declaring them (throw and throws) • Standard exception types (Throwables, Errors, Runtime Exceptions, checked exceptions) • Contracts (informal notion)
Collections and algorithms	• Overview: collection interfaces and implementations

	• Sample (1.5) classes (Lists and Maps) • Basic algorithms (e.g. sorting) and their complexity • Recursion and tree structures
Building simple GUIs	• Platform independent graphics and GUIs: AWT and Swing • Building basic GUIs • What's an applet - and what's it good for? • Handling events
Stream and file I/O	• Streams - System.out revealed • Text I/OFile handling • Options for storing data XML vs serialization vs. relational DB

Assessment:

50% programming exercises, 50% projects.

Assessment activity	**Length required**	**Weighting within unit**
Exercise 1 - Programming Activity	4 hours	10%
Exercise 2 - Programming Activity	10 hours	10%
Exercise 3 - Programming Activity	10 hours	10%
Exercise 4 - Programming Activity	10 hours	10%
Exercise 5- Programming Activity	10 hours	10%
Project Work - (Utilisation of skills learned to solve a two part problem)		
Project Work Part 1 - Programming Activity	25 hours	25%
Project Work Part 2 - Programming Activity	25 hours	25%

Further Information:

The course is delivered using a virtual learning environment (VLE). The material is supplied as a Computer Based training Package (CBT) and the students are guided through their study using a series of work packages presented by the VLE.
The CBT contains interactive self-assessment checks throughout the material. Each work package contains exercises to reinforce the learning process. These exercises can be submitted to the Course Assessor or Tutors who will respond with relevant feedback.
The VLE provides a bulletin board facility that the students are encouraged to use to communicate with one another, this board is moderated by the tutors and course assessor.
Each student is allocated to a tutorial group which meets on-line every week to discuss their study and any problems they are encountering. These

tutorials are led by a member of School staff with the appropriate skills. The tutorial content is logged and students who are unable to attend are able to access the logged material.

Prices & Bookings Information

COMP70180: Object Oriented Analysis and Design with UML

Course Duration:	16-17 weeks
Required Time per Week:	8 - 10 hours.
Course Contact:	John Sargeant (johns@cs.man.ac.uk)
Summary:	This course teaches essential skills in object-oriented analysis and design and the Universal Modelling Language. It is independent of particular software packages or programming languages, although there are a few small Java code examples. The only prerequisite is some familiarity with programming, not necessarily in an object-oriented language.
Prerequisites:	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++). 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 imperative languages listed above. Other than that, you will need to possess certain human qualities, such as the ability to think, discuss and experiment.

Objectives:

After successful completion of the module, a student will

- understand how to design software in an object-oriented manner.
- have mastered UML as a notation to support this design.
- have undertaken a reasonably sized OO design in UML as part of a team-work exercise.

Course Outline:

The course starts with a thorough introduction to object concepts, before explaining business modelling ("what do the customers need?") analysis ("what must the software do?") and design ("how will it do it?") Aspects of design covered include system and subsystem design and semi-formal specification of software responsibilities. No particular sofware development process is prescribed, but the approach taken is consistent with current best practice, in particular, the Rational Unified Process (RUP).

The course covers the OO software development life-cycle up to, but not including, the actual writing of code. The full syllabus is as follows:

Topic	**Content**
Object Overview	Objects; Classes; Inheritance; Object-Oriented Type Systems; Software Development Methodology; Engineering or Invention?; Artifacts of Object-Oriented Software Development; Classes, Responsibilities and Collaborators.
Requirements	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 Interface; Designing the Business Services; Thread Safety.
Specification	The Specification Process.

Assessment:

50% groupwork exercises, 50% online exam.

Further Details:

The course is delivered using a virtual learning environment (VLE). - Instruction takes the form of a written tutorial, delivered through the VLE. The tutorial includes a realistic case study to reinforce learning. Students are expected to study this tutorial in their own time.

At the end of each major topic, the students are asked to complete a substantial exercise; each exercise comes with its own guidelines. The exercises are based around another realistic case study. After the completion of each exercise, the course assessor will provide feedback to the students (via e-mail) and each will receive a copy of a model solution.

The VLE provides a bulletin board facility that the students are encouraged to use to communicate with one another, this board is moderated by the tutors and course assessor.

In this course students are split into project groups to develop solutions as a team (common practice in the work environment). The VLE supports the concept of teamwork, enabling the project groups to share information and to pass documents amongst the group. The tutors and course assessor can be contacted by e-mail or through the bulletin board for advice.

Each student is allocated to a tutorial group which meets on-line every week to discuss their individual study and any problems they are encountering. These tutorials are led by a member of School staff with the appropriate skills. The tutorial content is logged and students who are unable to attend are able to access the logged material.

Prices & Bookings Information

COMP70212: Self Timed Logic (Asynchronous Design)

Course Duration:	16-17 weeks
Required Time per Week:	8 - 10 hours.
Course Contact:	Stuart Anderson (stuart@cs.man.ac.uk)
Summary:	This course has been developed to give an understanding of the approaches required so that the designer is able to establish when it may be advantageous to use asynchronous techniques to solve a design problem.
Prerequisites:	Some background in digital design is assumed. In particular, it is assumed that concepts such as logic gates, flip-flops and Boolean logic are familiar.

Objectives:

On completion of this unit successful students will be able to:

- demonstrate an awareness of the potential advantages of asynchronous systems.
- understand asynchronous data and control protocols.
- be aware of asynchronous synthesis tools.
- show familiarity with the latest results from research into asynchronous systems.
- demonstrate an ability to write clear and concise reports on matters relating to asynchronous design.

Course Outline:

The full syllabus is as follows:

Topic	**Content**
Introduction	Why consider asynchronous circuits, aims and background, clocking versus handshaking.
Fundamentals	Handshake protocols, the Muller pipeline, delay models.
Static data-flow structures	Pipelines and rings, building blocks, example GCD.
Performance	A qualitative view of performance, quantifying

	performance, dependency graph analysis.
Handshake circuit implementations	The latch, Fork, join and merge, function blocks, mutual exclusion, arbitration and metastability.
Speed-independent control circuits	Signal transition graphs, synthesis procedure, Petrify, design examples using Petrify.
VLSI programming	Handshake circuits. an asynchronous HDL - Balsa. Using Balsa to describe circuits (buffers, stacks, recursive and parameterised structures).
An introduction to Amulet processors	Processor implemetation techniques, memory organization, asynchronous on-chip interconnect.

Further Details

Recommended Texts:

"Principles of Asynchronous Circuit Design – A Systems Perspective" Editors Jens Sparsø and Steve Furber, ISBN 0-7923-7613-7

Delivery Mode:

- Fully on-line distance learning with on-line support.
- 10 hours of on-line tutorials.
- Bulletin boards on-line.
- Work package documents supplied on-line.
- Course material – course notes and CBT package supplied on-line.
- On-line examination.
- Course book – paper based.

Prices & Bookings Information

COMP70300: Databases and Data Modelling

Course Duration:	16-17 weeks
Required Time per Week:	8 - 10 hours.
Course Contact:	Stuart Anderson (stuart@cs.man.ac.uk)
Summary:	The unit aims to introduce students to the fundamental concepts in databases, and is focused primarily on the database designer and application developer perspective, rather than on the implementation and technology aspects of database management systems. Students should expect to acquire both practical skills in database modelling, development, and query, along with an understanding of their theoretical underpinnings.
Prerequisites:	A understanding of mathematical 'Set Theory' and it's notation is required for this course.

Objectives:

The intended learning outcomes for this course as as follows:

Category of outcome	**Learning outcomes**
Knowledge and understanding	• Understand the fundamental concepts in data modeling, both at the conceptual and logical level, with specific reference to the ER and relational models, respectively; • Understand the design of database queries, starting from application-level data-intensive problems; • Understand the theoretical foundations of the data and query models, and use them to validate the soundness of a design solution; • Understand the concept of database transaction.
Intellectual skills	• Design a conceptual and a corresponding relational database schema from high level data requirements; • Verify the accuracy and completeness of a conceptual schema relative to user requirements, and the soundness of a relational schema; • Design database queries that solve specific application-level problems, and write them in SQL.
Practical skills	• Implement a database schema using a relational DBMS that supports standard SQL/DDL;

	• Implement and test SQL queries of various complexity on a relational DBMS, and embed them in a host application program; • Port the implementation to a different DBMS, by leveraging the independence of the logical schema from its implementation.

Course Content:

The unit begins with an introduction to the main rationale for the use of database systems, takes the student through a complete entity-relational (ER) modeling exercise, and then presents both the theoretical and practical aspects of the well-known relational data model. The gap between the models is bridged by introducing practical translation techniques, from ER to relational, along with a theory of relational schema normalization.

Following an exposition of relational algebra concepts, the SQL query language is then presented; this is complemented by additional modules on the use of SQL from within a host programming language, and on the concepts of database transactions.

The main example, a movies database, runs through the entire course; the reference DBMS for the unit is MySQL, which is used to implement the data model and to test the queries.

Assessment:

50% exercises, 50% projects.

Assessment activity	Weighting within unit
Five exercises worth 10% each	50%
A large course associated project	50%

Further Details:

Learning and Teaching Processes :

The unit is a 100% distance learning course, with the core material provided as a CBT package.The course is delivered using a virtual learning environment (VLE). The material is supplied as a Computer Based training Package (CBT) and the students are guided through their study using a series

of work packages presented by the VLE.
A series of assessments associated with the course must be periodically submitted to the Course Assessor or Tutors who will respond with relevant feedback.

The VLE provides a bulletin board facility that the students are encouraged to use to communicate with one another, this board is moderated by the tutors and course assessor. Students also have weekly online tutorials. Discussion of the assessments is encouraged, but the actual exercise and project work is done individually.

Prices & Bookings Information

MSEC40001: Entrepreneurial Commercialisation of Knowledge

Level:	MSc
Credit Rating:	15 credits (7.5 ECTS)
Degrees:	Advanced MSc
Pre-requisites:	None
Teaching period:	1 day per week (5 weeks)
Coursework and exercises:	10 days
Lecturer:	Dr. Martin Henery, Manchester Science Enterprise Centre
Start time:	Starts at 9am on the teaching days.
Limit on numbers:	50 participants

Introduction

This course unit is provided by the Manchester Science Enterprise Centre.

The unit provides an overview of the process that underpins the entrepreneurial journey from having an initial idea to the development of a succcessful venture. Particular emphasis will be given to the special role played by entrepreneurs in this process. An evaluation will be made of the particular skills, abilities and approaches that entrepreneurs are able to exploit when taking an idea forward into the marketplace. Students will be encouraged to reflect on their own strengths and how they may be used to best advantage in the process of moving ideas into the marketplace.

For more details of the Manchester Science Enterprise Centre consult the MSEC website.