Contents

Modules in Computer Science

From this session (1995-6), Computer Science courses are fully modularised and semesterised. The word module refers to courses offered under modularisation.

Module identifiers are of the form 'CS21020', where the third character indicates the level (which corresponds to the year in which the module is *usually* taken), and the last two characters indicate the credit weighting of the module. So, CS21020 is a level 2 module worth 20 credits.

Students must take modules worth a total of 120 credits in each year of study.

Modules which run entirely within the first semester will be assessed at the end of that semester. Modules which run entirely within the second semester, or those which run over both semesters, will be ssessed at the end of the second semester.

Students who started their degree course before modularisation (e.g., industrial year returners and MEng students) will have their degree assessed under the old regulations. In this respect, a 10 credit module is counted as a half unit, a 20 credit module as a full unit, etc. (All Computer Science level 3 modules count as starred units.)

The Computer Science department will consider *appropriate* modules from the Mathematics department or the Department of Information and Library Studies to be Computer Science units for the purposes of degree naming, so long as the total scheme remains a sensible one. Consult the departmental advisor if you need to know more.

To ensure best use of resources, the department reserves the right to withdraw modules, or otherwise amend the information given here.

A series of departmental seminars will be held during the year with invited speakers from industry, and this and other universities. The material in these seminars is regarded as examinable and questions involving knowledge of this material may be set on appropriate final year papers.

Students should note that an honours degree will not be awarded to candidates whose work on **CS39030** (the major project) does not reach honours standard. Similarly, MEng will not be awarded to students whose work on **CS49060** does not achieve honours standard.

BCS Exemption

Candidates for single honours BSc in Computer Science should note that, provided they achieve honours standard, they will be exempted from both parts of the British Computer Society's examinations and so will have fulfilled all the educational requirements for corporate membership of the Society and for Chartered Engineer status. Candidates who are successful in gaining a joint honours degree in Computer Science will be exempted from Part 1 of the Society's examinations.

Similar exemption has been provisionally granted for the engineering degrees (BEng and MEng) and for Computer Science with a European Language; this is due to be confirmed when the first students graduate.

Direct Entry to Part II

For direct entrants to Part II, a specially designed lead-in course is run during September. This is compulsory for all such students. Successful completion of this course is an *'exemption in Computer Science at Level 1'*. 4.1

Degree Schemes

On the following pages, the specific requirements for each degree scheme are described.

G500
BSc Computer Science

Students entering the second taught year

Students must take at least 80 credits in Computer Science and may take up to 120.
The normal choice of modules will include:

- **CS21020**, **CS22110**, **CS22210**;
- **CS15010** if they have not already done so.

CS23110, **CS26210**, **CS27310** and **CS28010** must be taken at some stage in the degree course. While they may be taken in the final year, it is advisable to taken them in the second unless there are strong arguments against it.

Students entering the third taught year

Students must take at least 80 credits in Computer Science and may take up to 120. At least 80 credits in Computer Science at level 3 are required for completion of the scheme.
The normal choice of modules will include:

- **CS33210**, **CS36210**, **CS37410**, **CS38010 CS39030**.

Core Modules

Before completion of this scheme, students must have studied the following modules in Computer Science:

Level 1	Level 2	Level 3
◯ **CS15010**	◯ **CS21020**	◯ **CS33210**
	◯ **CS22110**	◯ **CS36210** (or **CS26210**)
	◯ **CS22210**	◯ **CS37410**
	◯ **CS23110**	◯ **CS38010**
	◯ **CS27310** (or CS27220)	◯ **CS39030**
	◯ **CS26210** (or **CS36210**)	
	◯ **CS28010**	

G700
BEng Software Engineering

Students entering the second taught year

Students must take 120 credits in Computer Science (possibly including recognised modules from Mathematics). The normal choice of modules will include:

- **CS21020**, **CS22110**, **CS22210**, **CS22510**, **CS23110**, **CS23710**, **CS25010**, **CS26210**, **CS27310**, **CS28010**;
- **CS15010** if not already taken;
- one of the following if they require an additional module:**CS15110**, **CS25410**, **MA22110**, **MA22210**, **MA23110**, **MA23310**.

CS25410 must be taken at some stage in the scheme.

Students entering the third taught year

Students must take 120 credits in Computer Science.
The normal choice of modules will include:

- **CS33210**, **CS33310**, **CS34210**, **CS35010**, **CS36210**, **CS37410**, **CS38010**, **CS39030**;
- sufficient additional units chosen from the following. **CS32110**, **CS35410**, **CS36310**, **CS36410**, **CS37510**.

Core Modules

Before completion of this scheme, students must have studied the following modules in Computer Science:

Level 1	**Level 2**	**Level 3**
○ **CS15010**	○ **CS21020**	○ **CS33210**
	○ **CS22110**	○ **CS33310**
	○ **CS22210**	○ **CS34210**
	○ **CS22510**	○ **CS35010**
	○ **CS23110**	○ **CS36210** (or **CS26210**)
	○ **CS23710**	○ **CS37410**
	○ **CS25010**	○ **CS38010**
	○ **CS25410**	○ **CS39030**
	○ **CS26210** (or **CS36210**)	
	○ **CS27310** (or CS27220)	
	○ **CS28010**	

G701
MEng Software Engineering

Students entering the second taught year

Students must take 120 credits in Computer Science (possibly including recognised modules from Mathematics). The normal choice of modules will include:

- **CS21020**, **CS22110**, **CS22210**, **CS22510**, **CS23110**, **CS23710**, **CS25010**, **CS26210**, **CS27310**, **CS28010**;
- **CS15010** if not already taken;
- one of the following if they require an additional module: **CS15110**, **CS25410**, **MA22110**, **MA22210**, **MA23110**, **MA23310**.

CS15110 and **CS25410** must be taken at some stage in the scheme.

Students entering the third taught year

Students must take 120 credits in Computer Science.
The normal choice of modules will include:

- **CS25410**, **CS32110**, **CS33210**, **CS33310**, **CS34210**, **CS35010**, **CS36210**, **CS36310**, **CS36410**, **CS37410**, **CS38010**;
- **CS15110** if not already taken.

Students entering the fourth taught year

Students must take 120 credits with Computer Science.
The normal choice of modules will include:

- **CS43010**, **CS48110**, **CS49060**;
- remaining core Computer Science modules;
- sufficient other available Computer Science modules or appropriate Mathematics modules (**MA22110**, **MA22210**, **MA23110**, **MA23310**).

Core Modules

Before completion of this scheme, students must have studied the following modules in Computer Science:

Level 1	**Level 2**	**Level 3**	**Level 4**
CS15010	**CS21020**	**CS32110**	**CS43010**
CS15110	**CS22110**	**CS33210**	**CS48110**
	CS22210	**CS33310**	**CS49060**
	CS22510	**CS34210**	
	CS23110	**CS35010**	
	CS23710	**CS35410**	
	CS25010	**CS36210** (or **CS26210**)	
	CS25410	**CS36310**	
	CS26210 (or **CS36210**)	**CS36410**	
	CS27310 (or CS27220)	**CS37410**	
	CS28010	**CS37510**	
		CS38010	

G5T9
BSc Computer Science with a European Language

Students entering the second taught year

Students must take 80 credits in Computer Science and 40 credits in their chosen language.
The normal choice of modules will include:

- **CS21020**, **CS22110**, **CS22210**, **CS26210**, **CS27310**, **CS28010**;
- **CS15010** if they have not already done so.

Students entering the third taught year

Students must take 80 credits in Computer Science and 40 credits in their chosen language.
The normal choice of modules will include:

- **CS33210**, **CS36210**, **CS37410**, **CS38010**, **CS39030**;

Core Modules

Before completion of this scheme, students must have studied the following modules in Computer Science:

Level 1	Level 2	Level 3
◯ **CS15010**	◯ **CS21020**	◯ **CS33210**
	◯ **CS22110**	◯ **CS36210** (or **CS26210**)
	◯ **CS22210**	◯ **CS37410**
	◯ **CS26210** (or **CS36210**)	◯ **CS38010**
	◯ **CS27310** (or CS27220)	◯ **CS39030**
	◯ **CS28010**	

FG35, FG85, GG15, GG45, GGC5, GGD5, GN54
Joint Honours in Computer Science and Another Subject

Students entering the second taught year

Students must take at least 60 credits with Computer Science.
The normal choice of modules will include:

- **CS21020**, **CS22110** and **CS22210**.

Students entering the third taught year

Students must take at least 60 credits with Computer Science.
At least 60 credits in Computer Science at level 3 are required for completion of the scheme.
Students must take **CS33210**.

Core Modules

Before completion of this scheme, students must have studied the following modules in Computer Science:

Level 1	Level 2	Level 3
	◯ **CS21020**	◯ **CS33210**
	◯ **CS22110**	
	◯ **CS22210**	

G520
Information Science

Students entering the second taught year

In addition to modules required by the Department of Information and Library Studies students must take **CS22110**, **CS25010**, **CS26210** and **CS27310**.

Core Modules

Before completion of this scheme, students must have studied the following modules in Computer Science:

Level 1	Level 2	Level 3
	◯ **CS22110** ◯ **CS25010** ◯ **CS26210** ◯ **CS27310**	

4.1

Descriptions of Modules

The rest of this book provides details of each module offered by the Department at Level 2 and above. Level 1 modules are described elsewhere.

CS15010 Introduction to Computer Hardware

General Description

There is a very close relationship between system software design, system hardware design and the performance of the system as a whole. It is advantageous for computer scientists and software engineers to have an empathy for this relationship and therefore this module concentrates on the principles involved rather than fine detail. The module begins with a study of simple logic functions and how they can be combined to give increasing complexity and functionality. From this the course builds to a study of the operation of a typical system as a whole and the interaction between the various components within such a system. The laboratory work associated with the course is designed to consolidate the lecture material and gives practical experience in the use of various components and associated test equipment.

Commitment	Number of lectures	24
	Number of workshops	0
	Number of practicals	10 x 2 hours
Co-ordinator	Dr. John Hunt	
Other staff	Mr. Graham Parker	
Pre-requisites	None	
Co-requisites	**CS11010**	
Incompatibilities	None	
Assessment	Assessed coursework - 50%	
	Written exam - 50%	
Timing	This module is offered only in Semester 1	

Aims

To provide an appreciation of what computer hardware is, what it does, how it does it and how it influences the execution and performance of software.

- Introduction to logic and its hardware implementations.
- An appreciation of logic families.
- Familiarisation with small embedded systems.
- Familiarisation with interfacing such systems with the real world.
- An appreciation of technological advances in the hardware domain.

Objectives

On successful completion of this module, students will have a broad functional understanding of computer hardware. In addition, students will have detailed knowledge of:

- the function and application of combinational and sequential logic;
- the appropriate use of differing I.C. outputs;
- fundamental system components, their interactions and their dependencies;
- various memory implementations together with their suitability for given applications;
- the function and use of various peripheral interface devices;
- microprocessor evolution, together with an appreciation of how the area may develop in the future.

Syllabus

1. Number Systems and Logic — 1 lecture
   Decimal; Binary; Octal; Hexadecimal; Boolean Logic.
2. Combinational Logic — 2 lectures and 2 practicals
   Logic functions; Truth tables; Minterms; Boolean relationships; De Morgan's Theorem; Half Adders; Full Adders; Parallel Adders.
3. Sequential Logic — 2 lectures and 2 practicals
   RS Latch; Clocked devices; D-Type Latch; JK Flip-Flop; Storage registers; Counters; Shift registers; Logic representations.
4. Integrated Circuits - Chips — 2 lectures
   Packaging; Logic levels; Logic families; Totem Pole outputs; Open Collector outputs; Tri-State outputs; Comparators.
5. Basic Computer Architecture — 6 lectures and 6 practicals
   System components: Processor, RAM, ROM, Input/Output; Buses; Memory addressing; Memory maps; Line Decoders; I/O Mapping; Execution cycles; Bus timing; The Synchronous Bus; The Asynchronous Bus.
6. Interrupts — 1 lecture
   Application; Service routines; Priorities; Vectors; Power-On Reset.
7. Memory Types — 3 lectures
   Static RAM; Dynamic RAM; DRAM Controllers; Masked ROM; EPROM; EEPROM; Data Widths; PROM Decoding.
8. Peripheral Devices and Interfacing — 4 lectures
   Handshaking; Parallel Interfaces; Serial Interfaces; Timers; Watch Dog Timers; Direct Memory Access; Backplane Buses.
9. Micro-Processor Evolution — 3 lectures
   Micro-Processor Families; Motorola 68000 series; Intel 80X86 series; CISC and RISC; The Transputer; Single-Chip Processors; Parallelism; Branch Prediction; Pipelining; The Pentium; The PowerPC; The future?

Booklist

It is considered essential to purchase the following:

Alan Clements. *Principles of Computer Hardware*. Oxford, 1991.

Ian Sayers et. al. *Principles of Microprocessors*. CRC Press, 1991.

N.B. Only one of these books will be required by each student.

The following should be consulted for different approaches or for further information:

Alan Clements. *Microprocessor Systems Design*. PWS-Kent, 1992.

John Fulcher. *Introduction to Microcomputer Systems: Architecture and Interfacing*. Addison-Wesley, 1991.

V.C. Hamacher et, al. *Computer Organisation*. McGraw-Hill, 1989.

M. Rafiquzzaman et. al. *Modern Computer Architecture*. West, 1989.

2.7 A.J. van de Goor. *Computer Architecture and Design*. Addison-Wesley, 1989.

CS15110 Computer Architecture

General Description

This module builds on the material covered in **CS15010** providing a detailed study of the typical units constituting a microprocessor system. Emphasis is placed on Low-level language programming and associated software engineering issues together with microprocessor architecture and interfacing to the analogue and digital world. Semi-conductor technology is also covered at a materials level, together with a study of fabrication and design techniques. Practicals combine the assembly language programming aspects with the architecture and interfacing issues, to give a practical experience of the hardware/software interface.

Commitment	Number of lectures	24
	Number of workshops	-
	Number of practicals	10 x 2 hours
Co-ordinator	Dr. John Hunt	
Other staff	Mr. Graham Parker	
Pre-requisites	A grade C or better in GCSE Physics, Electronics, or Modular Science including modules on electricity *and* **CS15010**	
Co-requisites	None	
Incompatibilities	None	
Assessment	Assessed coursework - 50% Written exam - 50%	
Timing	This module is offered only in Semester 2	

Aims

The module aims to give an appreciation and experience of the interface between hardware and software along with the role of low-level languages. In addition, it is the intention that students will gain an understanding of the internal architecture and operation of microprocessor devices and the practicalities of interfacing to peripheral devices. An insight to Computer Aided Design environments will be provided by the practicals associated with full-custom design.

- To introduce assembly language programming.
- Raise awareness of the appropriate role of low-level languages.
- Highlight the intricacies and pitfalls of low-level language implementations.
- Application of established software engineering practices.
- To provide an understanding of the internal architecture of microprocessors in general.
- Give an awareness of the issues involved in interfacing to digital and analogue systems.

Objectives

It is not the objective of this module to qualify students as expert assembler language programmers or expert hardware engineers. Students will however, be expected to have a strong empathy toward the software/hardware /architecture interfaces, together with a sound practical experience of a range of typical devices and implementations. Specifically this will include:

- A comprehensive appreciation of the Motorola 68000 assembler language in terms of syntax, semantics and pitfalls.
- A detailed knowledge of a subset of the Motorola 68000 assembler language instruction set.

- An understanding of the importance of, and role of, a stack in the correct operation of a modular piece of code and as a parameter passing mechanism.
- An understanding of the use and application of bit-wise and byte-wise manipulation instructions.
- An understanding of the appropriate use of assembler language programming.
- A working knowledge of the internal registers of a typical microprocessor.
- A detailed knowledge of the Motorola 68230 PIT.
- The purpose and operation of Operational Amplifiers in the context of interface design.
- An understanding of ADC and DAC devices; Resolution; Linearity; Sampling rates; Aliasing; Resistor networks.
- An appreciation of the role and operation of Cache.
- A basic knowledge of semi-conductor fabrication.

Syllabus

1. INTRODUCTION — 1 lecture
   Course Organisation; Introduction to low-level languages; Characteristics of low-level languages; Processor characteristics.
2. MOTOROLA 68000 ASSEMBLER LANGUAGE — 9 lectures and 5 practicals
   Assembler language programmer's model - Register sets; Layout and format of code; Structure and design; Introduction to Instruction Set; Subroutines; Use of a stack; Role of Stack Pointer; Assembler directives; Control of thread of execution - Jumps and Branches; Role of the Status Register; Role of Programme Counter; Equivalent high-level language constructs; Bit manipulation; Addressing Modes; Applications; Subroutine workspace; Introduction to assembler environment; Use of environment and 68000 Tutor Boards.
3. PERIPHERALS AND INTERFACING — 6 lectures and 4 practicals
   68230 PIT: Detailed look at use of; Use of data sheets. Operational Amplifiers: Ideal characteristics; Virtual Earth; Buffers and Impedance matching; Comparators.
4. ARCHITECTURE — 3 lectures
   Internal Registers; Control Units: Hard-wired and Microcoded; Instruction types; Instruction cycles.
5. CACHE — 1 lecture
   Performance considerations; Fully associative cache; Direct mapped cache.
6. SEMI-CONDUCTOR TECHNOLOGY — 4 lectures and 1 practical
   Principles of CMOS technology; Materials; Doping; Fabrication of CMOS transistors; Fabrication of large scale devices; Design rule checking and relevance to computer scientists.

Booklist

It is considered essential to purchase the following:

Alan Clements. *Principles of Computer Hardware*. Oxford, 1991.

Ian Sayers et. al. *Principles of Microprocessors*. CRC Press, 1991.

N.B. Only one of these books will be required by each student.

The following should be consulted for different approaches or for further information:

Alan Clements. *Microprocessor Systems Design*. PWS-Kent, 1992.

Alan Clements. *Principles of Microprocessor Design*. PWS-Kent, 1992.

John Fulcher. *Introduction to Microcomputer Systems: Architecture and Interfacing*. Addison-Wesley, 1991.

King and Knight. *Programming the M68000*. Benjamin/Cummings, 1987.

Leventhal. *68000 Assembly Language Programming*. McGraw-Hill, 1986.

M68000 users manual. Motorola. 2.5

CS20010 Basic Computer Equipment

General Description

This is a basic introduction to the use of computers for any student in the faculty of science who has no significant previous experience of such equipment. It is particularly suitable for students in the School of Life Sciences. Students who complete the course successfully will be able to use personal computers and the campus mainframe for general purpose applications and have an appreciation of the range and capabilities of computer equipment. Intermediate level wordprocessing will be covered.

Commitment	Number of lectures	12
	Number of workshops	0
	Number of practicals	20 x 2 hours
Co-ordinator	Mr. Nigel Hardy	
Other staff	Not yet known	
Pre-requisites	None	
Co-requisites	None	
Incompatibilities	Not available to students who have significant experience of using computers. **CS10010**	
Assessment	Assessed coursework - 100%	
Timing	This module is offered only in Semester 1	

Aims

This module will:

- introduce students to potential benefits of information technology in their academic lives;
- introduce them to the campus computing facilities and their use;
- allow them to develop skills and confidence in general computer usage;
- begin the development of an understanding of when and how computers can and cannot assist effectively.

Objectives

On successfull completion of this module, students will have an understanding of:

- the hardware components found in typical computer systems and broadly what their functions are;
- the typical cost and performance characteristics of a range of technologies and products;
- the major benefits of using computer technology to assist with document product;
- the basic functionality of spreadsheets and databases.

On successful completion of the module students should be able to:

- use Microsoft Windows to launch applications and handle multiple applications at once;
- access a command string interpreter interface to a Computer Unit UNIX machine and perform a small range of simple tasks;
- create and maintain a personal directory structure;
- produce effective and well designed documents of significant complexity using Microsoft Word;
- create graphics (using bit-mapped and object based packages) and charts and include them in wordprocessed documents;
- create and use simple spreadsheets;

- create and use simple, single table databases at a ‘cardfile’ level;
- use electronic mail;
- use on-line facilities such as the CD-ROMS and Internet services;
- evaluate and select appropriate equipment for a range of non-technical applications; this to include software and display, printing and storage devices.

Syllabus

1. INTRODUCTION — 1 lecture
   Course organisation. Aims and Objectives.
2. SURVEY OF SOFTWARE — 2 lectures
   Operatings systems; user interfaces, utilities; applications programs; user written programs.
3. SURVEY OF HARDWARE — 2 lectures
   Processors; primary store; secondary store; displays; printers; pointing devices; digitisers.
4. NETWORKING AND COMMUNICATIONS — 1 lecture
   Range of technology and bandwidth; campus network; Janet, Super Janet and the Internet.
5. GRAPHICS APPLICATIONS — 1 lecture
   "Draw", "Paint", charting and image packages; the bit-mapped and object based storage techniques and their implications.
6. DATABASES — 2 lectures
   Tables, records and fields; validation; the need for and complexities of multiple table databases.
7. SPREADSHEETS — 1 lecture
   Practical examples of more powerfull facilities; macros; available functions.
8. DATA PROTECTION LEGISLATION — 1 lecture
   The Data Protect Act and its implications.
9. ADVANCED APPLICATIONS — 1 lecture
   The concept of artificial intelligence; its historical development; the application of expert systems.
10. MICROSOFT WINDOWS — 1 practical
    General use and manipulaton of Windows; launching applications.
11. E-MAIL; FILES I. — 1 practical
    Creation, manipulation and maintenance of a directory structure.
12. WORD PROCESSOR — 4 practicals
    General use of Microsoft Word.
13. FILES II — 1 practical
    Revision of the use of directory structures following practical experience of them.
14. GRAPHICS — 2 practicals
    Bit mapped and object orientated graphics applications (Mircrosoft Paint and Draw).
15. SPREADSHEETS — 2 practicals
    A brief introduction to the potential of spreadsheets (using Microsoft Excel).
16. MAINFRAME USE — 3 practicals
    Logging on to campus mainfarmes; simple use of a UNIX shell; the UNIX filestore; I/O redirection; simple editing.
17. DATABASES — 2 practicals
    Experience of setting up a simple database; search and report generation.

18. ASSIGNMENTS 4 practicals
Time is allocated for the work on assignments.

Booklist

The following should be consulted for different approaches or for further information:

Nancy B. Stern and Robert A. Stern. *Computing in the Information Age*. John Wiley and Sons Inc, New
4.1 York, 1993.

CS20210 Computer Programming for Life Sciences

General Description

This module introduces the principles of procedural computer programming. It is intended for students doing degrees in life sciences but may be taken by others. On successful completion of the course students will be able to design and implement simple programs in a particular high level language. They will also have an appreciation of the software development process which would enable them to evaluate new software as a potential tool in their work and to liaise meaningfully with software developers. Other programming paradigms will be briefly introduced.

Commitment	Number of lectures	24
	Number of workshops	0
	Number of practicals	0 (Students are expected to spend unscheduled time on the significant practical aspects of this course)
Co-ordinator	Mr. Nigel Hardy	
Other staff	Not yet known	
Pre-requisites	**CS20510** or equivalent experience	
Co-requisites	None	
Incompatibilities	CS10210, CS10310, **CS11010**, **CS21020**	
Assessment	Assessed coursework - 50% Written exam - 50%	
Timing	This module is offered only in Semester 1	

Aims

Through an introduction to the software life cycle; a study of the principles of procedural programming; examination of the details of one high level language; experience of building small programs; consideration of relevant case studies and examination of procedural languages in a wider context this course will:

- give students an insight into the process of building software;
- allow them to assess the efficacy of building new software as a potential solution to application problems in their own discipline
- allow them to work more effectively with and to modify and customise existing software;
- give them the skills to build small items of software.

Objectives

On successful completion of this course, students will have an understanding of:

- the size and importance of the stages in simple software life cycles;
- the elements of procedural programming languages;
- the resources necessary to build new software.

They will be able to:

- use a development environment available at Aberystwyth;
- develop small programs of reasonably high quality.

Syllabus

1. INTRODUCTION — 1 lecture
   A practical overview of machines and instruction sets; high level languages; interpretation and compilation; data and algorithms.

2. NOTATION AND AN INTRODUCTORY CASE STUDY — 1 lecture
   Syntax and semantics; lexical conventions; reserved words and identifiers. A small code example will be considered from these aspects.

3. THE DEVELOPMENT ENVIRONMENT — 1 lecture
   Practical details of using the chosen environment.

4. SIMPLE DATA TYPES AND VARIABLE DECLARATION — 1 lecture
   The concept of type; constants; the concept of a variable; the need for a value of data declarations.

5. BASIC CONTROL STRUCTURES — 5 lectures
   The need for control constructs; basic branching and looping; the need for and form of boolean expressions; control construct syntax. Subprograms as control constructs; procedures and functions.

6. A CASE STUDY IN CONTROL STRUCTURES — 1 lecture
   A structured program to demonstrate the benefits of control structures.

7. DATA STRUCTURES — 4 lectures
   The need for compound data structures; arrays and records. User defined types; enumerated types. Simple data structure design.

8. A FULL CASE STUDY IN CODING — 1 lecture
   A case study covering all aspects of coding covered to date.

9. PROGRAM DESIGN — 2 lectures
   The need for a design phase. Simple design aids; flow diagrams; pseudocode.

10. ESTABLISHMENT OF FUNCTIONAL REQUIREMENTS — 1 lecture
    The need for a requirements establishment phase. Examples of poor definition. The difficulties of establishing requirements.

11. TESTING — 1 lecture
    The need for testing; test strategies and plans. Examples of testing schemes.

12. THE SOFTWARE LIFE CYCLE — 1 lecture
    A review of the stages met, in context. The relative importance of the stages in projects of different types.

13. A MAJOR CASE STUDY — 4 lectures
    The interactive development of a program of significant complexity. Example requirements, design and testing documents.

Booklist

Students are likely to need ready access to the following:

Walter J. Savitch. *An Introduction to the Art and Science of Programming*. Benjamin/Cummings Pub.
4.1 Co, Redwood City, Calif, 3rd edition, 1991.

CS20510 Computing for Life Sciences

General Description

This module is intended primarily for students taking degrees in Life Sciences. The function, value and effective use of spreadsheets and simple databases will be covered. The availability and use of network information resources will be explored.

The benefits of simple interfacing for environmental monitoring and for the control and monitoring of experiments will be outlined and the basic technology and techniques of such interfacing will be introduced.

Commitment	Number of lectures	24
	Number of workshops	0
	Number of practicals	10 x 2 hours
Co-ordinator	Mr. Nigel Hardy	
Other staff	Not yet known	
Pre-requisites	**CS20010** or equivalent experience	
Co-requisites		
Incompatibilities	**CS10010**, **CS10510**, **CS10610**, **CS10810**, **CS21020**	
Assessment	Assessed coursework - 50%	
	Written exam - 50%	
Timing	This module is offered only in Semester 2	

Aims

This module aims to increase general familiarity with and appreciation of the potential of computing techniques to Life Scientists. The following four particularly relevant areas or technology are considered:

- spreadsheets for general purpose use and for model building;
- databases;
- network information resources;
- interfacing techniques for sampling and experimental control.

Objectives

On successful completion of the module students will be able to:

- use most facilities of a modern spreadsheet, including charting and macro recording;
- recognise and analyse potential applications of database technology;
- perform data design for simple relational databases;
- recognise a number of potential problems in implementing such a database;
- set up tables, relationships, queries and reports using Microsoft Access;
- search for and retrieve information available on the Internet;
- recognise the potential for using microcomputer technology to monitor and sample simple sensor data during observation, trials or experiments both in the laboratory and in the field;
- recognise the potential for using computers to control experimental systems.

Syllabus

1. SPREADSHEETS — 4 lectures and 3 practicals
   Revision of basic operation; complex expressions; conditionals; lookup tables; advanced charting; macros. The exploration and guided extension of a provided Microsoft Excel spreadsheet.
2. SINGLE TABLE DATABASES — 3 lectures
   Uses; examples. Definitions - table, record, field. Queries, forms and reports.
3. USE OF MICROSOFT ACCESS — 1 lecture and 3 practicals
   Practical explanation and guidance. Creation of a simple single table database from a provided design; entry of data; searching and report generation.
4. MULTI-TABLE DATABASES — 4 lectures and 3 practicals
   The need for multiple tables. Entities, attributes, relationships, E-R modelling. Domain key normal form. Examples. Exploration and guided enhancement of a provided multi-table database.
5. FREE TEXT AND HYPERTEXT — 1 lecture
   Searching and browsing. The principles of hypertext production. Example systems in Life Sciences.
6. ON-LINE INFORMATION RESOURCES — 4 lectures and 1 practical
   CD-ROM access and use. The Internet; ftp, news, gopher and WWW operation; obtaining information; searching for information. Exploration of on-line resources as available.
7. SIMPLE COMPUTER COMMUNICATIONS — 3 lectures
   Serial and parallel communications; standards; configuration parameters; data rates; applications. Local and wide area networks; data rates; applications.
8. INTERFACING — 4 lectures
   The analog/digital distinction; converters and their characteristics; example applications.

Booklist

The following should be consulted for different approaches or for further information:

David M Kroenke. *Database Processing - Fundamentals, Design, and Implementation*. Prentice Hall International Editions, 5th. edition, 1995. ISBN 0-13-320128-7.

Nancy B. Stern and Robert A. Stern. *Computing in the Information Age*. John Wiley and Sons Inc, New
4.2 York, 1993.

CS21020 Program Design, Data Structures and Algorithms

General Description

This module builds on the foundations of the first year modules on program design and provides a thorough grounding in the design of data structures and algorithms and an introduction to the basic concepts of object-oriented design.

Commitment	Number of lectures	48
	Number of workshops	8
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
Other staff	Not yet known	
Pre-requisites	Pass or exemption in Computer Science at Level 1	
Co-requisites	None	
Incompatibilities	None	
Assessment	Assessed coursework - 50%	
	Written exam - 50%	
Timing	This module extends over both Semester 1 and Semester 2	

Aims

This module provides an introduction to data structures and their use in solving programming problems. The course emphasises the use of abstract data types and the contribution that abstraction and encapsulation can make to the comprehensibility, reusability and robustness of programs. Ada is used as the main language of expression with the intent of providing a means of allowing the student to naturally express these design objectives in code.

As well as providing a solid grounding in the major data structures and algorithms of Computer Science, the course stresses the development of problem solving skills through a number of programming assignments.

Objectives

On successful completion of this module, students should:

- understand the principles of abstraction and encapsulation as they apply to the design of abstract data types and programs;
- understand the importance of the running time behaviour of algorithms and how this is expressed and determined;
- have a familiarity with some of the main approaches to algorithm design such as greedy algorithms, divide and conquer and dynamic programming;
- be able to evaluate and choose appropriate data structures and algorithms for a range of programming problems;
- be able to design and implement significant programs in Ada;
- have an appreciation of the importance of program correctness and some of the techniques used to ensure it.

Syllabus

1. INTRODUCTION TO COMPLEXITY — 4 lectures
   O() notation, growth rates. Measurement of execution time of some real programs and estimation of their time complexity. Some examples of time/space trade-offs.

2. RE-INTRODUCTION TO ABSTRACT DATA TYPES — 4 lectures
   Issues of correctness as they relate to the definition of ADTs. The key ideas of abstraction and encapsulation. Notations for describing ADTs. Ada support for their implementation: packages, exceptions and generics.

3. INTRODUCTION TO POINTERS AND DYNAMIC STORAGE ALLOCATION AND THE ADVANTAGES AND DISADVANTAGES COMPARED WITH STATIC ALLOCATION — 4 lectures
   Failure modes associated with heap allocation. An examination of a familiar ADT implemented using dynamic storage. Abstract specification versus concrete implementation.

4. STORING AND RETRIEVING DATA BY KEY (1) — 8 lectures
   This problem will be used to motivate the discussion of a wide variety different implementation techniques. The features of some typical solutions will be related to the dimensions of the problem such as the volume of data to be handled, volatility and the operations required. Internal Storage: linear and binary searching. A brief introduction to proving programs correct with particular reference to the use of loop invariants. Linked representations; an introduction to hashing and binary search trees.

5. STORING AND RETRIEVING DATA BY KEY (2): EXTERNAL STORAGE — 6 lectures
   Performance issues. Using Ada Direct_IO to implement persistent random access storage. Hashing and B-tree organisations.

6. REPRESENTING TEXT — 4 lectures
   A simple text editor will be used as the basis of a substantial design case-study showing how the design decisions concerning the different ways of representing text affect performance and ease of implementation. String matching algorithms and their performance (for the searching operations provided by the editor).

7. SORTING — 4 lectures
   Divide and conquer algorithms. An introduction to proving the performance of an algorithm.

8. REPRESENTING COMPLEX RELATIONSHIPS: GRAPHS AND THE ALGORITHMS USED TO PROCESS THEM — 8 lectures
   Some examples of greedy algorithms. Terminology and implementation considerations. A look at some graph-related problems such as: finding a route (shortest paths); planning a communications network (minimum spanning trees); network routing management (flow graphs); compiling a program or planning a project (topological sorting).

9. LIMITATIONS OF ADTS — 4 lectures
   What happens when a reusable component does almost what you want. An introduction to object-oriented programming and design and the object-oriented features of Ada 95.

10. DESIGNING ADTS, CLASSES AND OBJECTS IN OTHER LANGUAGES — 4 lectures
    A brief examination of C++.

Booklist

Students are likely to need ready access to the following:

D. F. Stubbs and N. W. Webre. *Data Structures with Abstract Data Types and Ada*. PWS-Kent, 1993.

Mark Allen Weiss. *Data Structures and Algorithm Analysis in Ada*. Benjamin/Cummings, Redwood City, California, 1993.

N.B. Students will probably wish to choose one of the above.

The following should be consulted for different approaches or for further information:

Alfred Aho, John Hopcroft, and Jeffrey Ullman. *Data Structures and Algorithms*. Addison-Wesley, Reading, Massachusetts, 1983.

Alfred Aho and Jeffrey Ullman. *Foundations of Computer Science*. Computer Science Press, New York, 1992.

Grady Booch and Doug Bryan. *Software Engineering with Ada*. Benjamin/Cummings, Redwood City, California, 3rd. edition, 1994.

Timothy A. Budd. *Classic Data Structures in C++*. Addison-Wesley, 1994.

Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, Massachusetts, 1990.

Robert L Kruse. *Data Structures and Program Design*. Prentice-Hall, Englewood Cliffs, New Jersey, second edition, 1987. (The first edition is also available from the library).

Robert Sedgewick. *Algorithms*. Addison-Wesley, Reading, Massachusetts, 1988.

Thomas A Standish. *Data Structures, Algorithms and Software Principles*. Addison-Wesley, Reading, Massachusetts, 1994.

Rebecca Wirfs-Brock, Brian Wilkerson, and Lauren Wiener. *Designing Object-Oriented Software*. Prentice Hall, 1990.

N.B. The book by Wirfs-Brock *et. al.* provides a useful introduction to the concepts of object-oriented programming and design which are covered later in the course. Budd shows how many of the data structures covered in the course can be implemented using C++. 4.1

CS22110 The Software Development Lifecycle

General Description

The objectives of the lecture course are first to introduce students to the best traditional practices for the specification, design, implementation, testing and operation of large software systems; and second to provide a framework for the more detailed material on design which is taught in other courses. The practical work is a group project.

Long thin module. No lectures in Semester 2

Commitment	Number of lectures	18
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
Other staff	Not yet known	
Pre-requisites	Pass or exemption in Computer Science at Level 1 or CS10310 by arrangement with the department	
Co-requisites	None	
Incompatibilities	None	
Assessment	Assessed coursework - 100%	
Timing	This module extends over both Semester 1 and Semester 2	

Aims

This module aims to introduce students to the basic principles of software engineering and to give them experience of developing a software system in a team. Specifically, it aims to:

- introduce students to the best traditional practices for the specification, design, implementation, testing and operation of large software systems;
- provide a framework for the more detailed material on design which is taught in other courses;
- involve students in the development of a piece of software which approximates as closely as possible in the university environment the software development conditions found in industry.

Objectives

On successful completion of this module students should:

- understand the approach and obligations of the professional engineer and the analogies between software and other branches of engineering;
- have an appreciation of the software life cycle, and the management issues involved in software development;
- be aware of the need for quality assurance and know how it is applied in the software industry;
- understand the concept of configuration management and be able to use the UNIX tools sccs and make;
- be able to produce the key deliverables of the software life cycle;
- have an appreciation of the use of CASE tools in software engineering.

Syllabus

1. INTRODUCTION — 1 lecture
   The approach and the obligations of the professional engineer. Software as an engineering artifact. Analogies between software and other branches of engineering.

2. THE SOFTWARE LIFE CYCLE — 2 lectures
   Description of the phases of the software life cycle and the major deliverables and activities associated with each phase.

3. PROJECT MANAGEMENT — 2 lectures
   Planning and cost estimation. Progress monitoring. Team structure and team management.

4. QUALITY MANAGEMENT — 3 lectures
   Validation, verification and testing. Quality plans. Walkthroughs, code inspections and other types of review. Role of the quality assurance group. Standards (international, national and local).

5. CONFIGURATION MANAGEMENT — 2 lectures
   Baselines. Change control procedures. Version control. Software tools to support configuration management: 'sccs' and 'make'.

6. REQUIREMENTS SPECIFICATION — 2 lectures
   The IEEE standard for requirements specifications. Validation of requirement by e.g., prototyping. Deficiencies in the traditional approach to requirements.

7. DESIGN — 2 lectures
   Outline (architectural) design and detailed design. Use of abstraction, information hiding, functional and hierarchical decomposition at levels higher then the individual program. Contents of design documentation. [Note that both **CS23110** and CS27220 contain more detailed coverage of design methods.]

8. IMPLEMENTATION AND TESTING — 2 lectures
   Importance of suitable programming languages and a good support environment. Testing strategies. Testing tools: static and dynamic analysers, test harnesses and test data generators, simulators. Performance testing. Regression testing. User documentation and training. Cutover. Post-implementation reviews.

9. TOOLS — 2 lectures
   How CASE tools can aid the software engineer. Upper and lower CASE. Meta CASE tools. The capability maturity model.

Booklist

It is considered essential to purchase the following:

Ian Sommerville. *Software Engineering*. Addison Wesley, 4th edition, 1992.

The following should be consulted for different approaches or for further information:

V. Berzins and Luqi. *Software Engineering with Abstractions*. Addison Wesley, 1991.

S.L. Pfleeger. *Software Engineering*. Macmillan, 2nd edition, 1991.

Ian Sommerville and R. Morrison. *Software Development with Ada*. Addison Wesley, 1987. This book includes material which students will find useful in connection with the design phase of the group project.

H. van Vliet. *Software Engineering: Principles and Practice*. John Wiley and Sons, 1993. 4.2

CS22210 Operating Systems and Concurrency

General Description

This course introduces the basic principles of operating systems and discusses the important underlying concepts, including the problems of concurrency.

Commitment	Number of lectures	24
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
Other staff	Not yet known	
Pre-requisites	None	
Co-requisites	**CS21020**/C210	
Incompatibilities	None	
Assessment	Written exam - 100%	
Timing	This module is offered only in Semester 1	

Aims

The purpose of this module is to provide a basic introduction to operating systems and the underlying concepts of processes, resource management, security, and concurrent systems. Software engineers should be aware of techniques associated with these concepts which are needed in applications other than operating systems.

Objectives

At the completion of this module a student should:

- understand enough about operating systems to be able to use them effectively;
- be familiar with the concepts of resource management, especially of processes, memory and filestore;
- understand the problems that arise in concurrent systems, and a range of techniques for tackling them;
- have an appreciation of the issues involved in security of computer systems.

Syllabus

1. INTRODUCTION TO OPERATING SYSTEMS — 3 lectures
   A brief historical overview with examples of important systems. Different styles of operating system (e.g. real time, batch, program development). Operating systems as a vehicle for the portability of application programs. The gross structure of an operating system: layered architecture and its benefits for implementation and maintenance.
2. PROCESSES AND CONCURRENCY — 6 lectures
   The idea of a process and its life history. Process control blocks, scheduling. Co-operating processes: exclusion, synchronisation, inter-process communication. Semaphores, monitors, messages. Multi-tasking in Ada, the rendezvous.
3. RESOURCE MANAGEMENT — 6 lectures
   Different types of resources. Deadlock prevention and avoidance. Memory management. Segmentation, paging. Discard algorithms, thrashing. Associative stores.
4. PROGRAM AND HUMAN INTERFACES — 1 lecture
   Operating system services. Human interface, command languages.

5. INPUT-OUTPUT — 3 lectures
   Input-output services. Filing systems, disc space management.

6. DISTRIBUTED SYSTEMS — 3 lectures
   The need for distributed systems, tightly-coupled and loosely coupled systems, the problems which they present to the operating system.

7. SECURITY — 2 lectures
   Discretionary and mandatory security. User authentication. Encryption. System management.

Booklist

Students are likely to need ready access to the following:

H.M. Deitel. *An Introduction to Operating Systems*. Addison Wesley, 2nd. edition, 1990.

Abraham Silberschatz and Peter B. Galvin. *Operating system concepts*. Addison-Wesley, Reading, Mass, 4th. edition, 1994.

N.B. Students will probably wish to choose one of the above.

The following should be consulted for different approaches or for further information:

M.J. Bach. *The Design of the UNIX Operating System*. Prentice Hall, 1986. Of value to students interested in the design of the UNIX Operating System. 4.1

CS22510 Case Studies in System Design

General Description

This module considers the complexity of large software systems through the detailed study of three such systems exemplifying a wide range of problems. The process of engineering design as it is applied in these systems is investigated and students are given some experience of the problems of maintaining substantial software systems.

Commitment	Number of lectures	24
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
Other staff	Not yet known	
Pre-requisites	**CS22210**/C220	
Co-requisites	**CS21020**/C210, **CS22110**/C220	
Incompatibilities	None	
Assessment	Assessed coursework 20% Written exam - 80%	
Timing	This module is offered only in Semester 2	

Aims

This module aims to give students insight into the problems involved in the design and implementation of large and complex software systems by presenting actual experience from three case studies.

Objectives

On successful completion of this module students should:

- have an understanding of the complexity of large software systems;
- be familiar with the process of engineering design as it is applied in software systems;
- have some experience of the problems of maintaining substantial software systems;
- have experience of testing such systems.

Syllabus

1. INTRODUCTION — 1 lecture
   Overview of the module and brief outline of the three case studies.
2. CASE STUDIES — 3 x 7 lectures
   About seven lectures will be given on each of three different case studies. The lectures will cover, as appropriate, the real world situation to be modelled, the requirements specification, the design and the testing of each of the systems. A brief overview of the implementation environment will also be given where necessary.
3. SUMMARY AND CONCLUSIONS — 1 lecture
   A high level view of lessons learned from all three case studies and the assignment.

Booklist

4.1 Not appropriate.

CS23110 Small Real-Time Systems

General Description

A small real-time system is a computer which directly controls a larger system, in which it is embedded. This structure, with direct interaction beween the computer and its physical environment, contrasts with conventional uses of computers through the distinctive hardware-software interaction and multiple control loops. There are special problems for designers and implementers, which the course addresses.

Commitment	Number of lectures	24
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
Other staff	Not yet known	
Pre-requisites	**CS15010**, **CS22210**/C220	
Co-requisites	**CS21020**/C210	
Incompatibilities	None	
Assessment	Assessed coursework - 20% Written exam - 80%	
Timing	This module is offered only in Semester 2	

Aims

To introduce students to the ideas of embedded systems (through studying small microprocessor-based systems for a wide range of application areas), and to the techniques used for developing them (including such systems for safety-related applications).

Objectives

A student who successfully completes the course should be able to:

- identify the wide, and increasing, use of small embedded systems in domestic, office, transport and industrial process control applications;
- specify the desired behaviour of an embedded system, including its real-time operation and quality requirements;
- select appropriate devices, both analogue and digital, that are used for input and output in embedded systems, with their interfaces;
- select an appropriate hardware and software design taking account of their joint influence on system performance and dependability;
- define required characteristics of multi-tasking software, for use with real-time executive or kernel;
- recognise their strengths and limitations of methods such as MASCOT and HOOD for the design of real-time systems, and associate tools;
- design a small real-time system in MASCOT;
- describe an appropriate standard for developing and checking a safety-related system with justifiable confidence;
- discuss the implications and risks of using an embedded system in a safety-related application.

Syllabus

1. INTRODUCTION — 2 lectures
   Scope and definitions. Examples.
2. RESPONDING TO THE REAL WORLD — 4 lectures
   Hardware and software interaction. Input/output devices and interfaces. Interrupts and polling; control strategies. Hardware-software tradeoffs; architectures for real-time.
3. SOFTWARE STRUCTURE — 4 lectures
   Application tasks and real-time kernels. Scheduling; resource allocation. Reliability: causes and effects of mistakes. Relationship between design and programming language.
4. DESIGN PRINCIPLES AND PRACTICE — 7 lectures
   What a design method can and cannot give. MASCOT (Modular Approach to System Construction, Operation and Testing). Example and use of MASCOT. HOOD (Hierarchical Object Oriented Design). Example in HOOD. Impact of performance issues on design.
5. IMPLEMENTATION PRINCIPLES AND PRACTICE — 3 lectures
   Tools to assist development. Host-target development, simulation, emulation. Debugging aids, performance monitors.
6. CHECKING PRINCIPLES AND PRACTICE — 3 lectures
   Validation and Quality Assurance. Safety related systems and related standards.
7. REVISION AND CONCLUSION — 1 lecture

Booklist

Students are likely to need ready access to the following:

A. Burns and A. Wellings. *Real-time Systems and their Programming Languages*. Addison Wesley, 1990.

J. E. Cooling. *Software Design for Real-time Systems*. Chapman and Hall, 1991.

Caxton C. Foster. *Real-Time Programming - Neglected Topics*. Addison-Wesley, 1981.

The official handbook of MASCOT : Version 3.1. Royal Signals and Radar Establishment, 1987.

Ian C. Pyle. *Developing Safety Systems*. Prentice Hall International, 1991.

N.B. The book by Cooling is highly recommended but rather expensive.

The following should be consulted for different approaches or for further information:

A. Amghar. *Microprocessor System Development*. Prentice Hall, 1990.

Andrews. *Concurrent Programming*. Benjamin Cummings, 1991.

S. Bennett and G.S. Virk, editors. *Computer Control of Real-time Processes*. Peter Peregrinus, 1990.

Philip McDowell. *Choosing and Using 4 bit Microcontrollers*. Newnes, Oxford, 1994.

Ian Pyle, Michel Lissandre, Peter Hruschka, and Ken Jackson. *Real-Time Systems - Investigating Industrial Practice*. Wiley, 1993.

Ken Shumate and Marilyn Keller. *Software Specification and Design - A Disciplined Approach for
4.1 Real-Time Systems*. Wiley, 1992.

CS23710 C and UNIX Systems Programming

General Description

The main objectives of this course are to teach the basic features of the C programming language, the structure of the Unix operating system and the Unix interface to the programmer.

The facilities of C will be introduced by comparison with those of Ada, the emphasis being on the rationale behind the language design decisions for the two programming languages. Emphasis will also be placed on C programming incorporating good modular design, readability, portability, and use of appropriate tools.

The second part of the course will address the structure of the Unix operating system and the interface presented by functions and system calls for use from C.

The module will also consider the relevance of standards, both in C and in Unix.

Commitment	Number of lectures	24
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
Other staff	Not yet known	
Pre-requisites	**CS22210**/C220	
Co-requisites	**CS21020**/C210	
Incompatibilities	None	
Assessment	Assessed coursework - 100%	
Timing	This module is offered only in Semester 2	

Aims

The aim of this course is to teach the student the fundamentals of the C language. In order to compare the facilities and underlying philosophy of C with that of Ada, the first part of the course will draw closely on the material that is covered in **CS21020**.

The UNIX operating system is studied as an example of the operating systems material covered in **CS22210** and also as an application of the C language.

- The first part of the course will cover the C language, paying special attention to its differences in facilities and philosophy to those of Ada.
- The second part of the course looks at the UNIX operating system as an example of an operating system programming interface and also at the binding that the UNIX operating system presents to the C programmer.

Objectives

The detailed objectives of this module are:

- To provide students with enough facility in C programming that they are able to read, write and debug C programs.
- To examine the philosophy behind the creation of the C language.
- To examine the similarities and differences between C and Ada.
- To examine the structure of the UNIX operating system.
- To examine the philosophy behind the creation of UNIX.
- To examine the programming interface to UNIX from C.
- To give the student practice in understanding operating system facilities.
- To underline the importance of standards.

Syllabus

1. BASIC CONCEPTS — 1 lecture
   History of the C language, philosophical differences between C language design and Ada. Basic form of a C program compared with that of an Ada program. Using the compiler.
2. CONTROL STRUCTURES — 2 lectures
   Sequence, branching and iteration in C compared with that of Ada.
3. BASIC DATA STRUCTURES — 1 lecture
   Review of basic data types and operators in C.
4. SUBPROGRAMS — 1 lecture
   Discussion of ways in which functions are implemented, and used in C, including parameter passing mechanisms.
5. COMPOSITE DATA STRUCTURES — 1 lecture
   A first discussion of Arrays in C. Input/Output.
6. SOFTWARE SUPPORT TOOLS — 1 lecture
   Make, Lint, Debuggers (UPS). Libraries and library utilities.
7. C PROGRAMMING STYLE AND PORTABILITY — 1 lecture
   Language standards. Portability. Programming standards.
8. ARRAYS, POINTERS AND FUNCTIONS — 2 lectures
   A discussion of pointer data types, how they relate to arrays, and how they contrast with the Ada Access type.
9. DYNAMIC DATA STRUCTURES — 2 lectures
   Implementation of various record structures and dynamic structures. Pointers. Malloc. Examples in C. Parallels will be drawn with similar constructs in Ada.
10. PITFALLS — 1 lecture
    Major problem areas. Design rationale of C and of Ada in problem areas.
11. FURTHER FEATURES — 1 lecture
    C preprocessor, header files, conditional inclusion, macro substitution, bitwise operators, casts, enumeration, scope, static and external declarations, separate compilation
12. THE STRUCTURE OF UNIX — 2 lectures
    An overview of the structure of the UNIX operating system and a description of the major components, including: the kernel, device drivers, the filesystem, the process, the terminal. An introduction to the programming interface to Unix.
13. THE FILESYSTEM AND FILES — 2 lectures
    The structure of the filesystem and associated operations. The types of file and the operations on files.
14. THE PROCESS AND INTERPROCESS COMMUNICATION — 2 lectures
    The concept of the process and associated operations.
15. THE TERMINAL — 1 lecture
    A study of the general interface.
16. NETWORK PROGRAMMING — 1 lecture
    A study of network programming using remote procedure calls and the Transport Level Interface.
17. STANDARDS — 1 lecture
    ANSI 'C'; IEEE POSIX; UNIX International SVID; XOPEN XPG; COSE
18. WINDOW SYSTEMS AND EVENT-DRIVEN PROGRAMMING — 1 lecture
    An appreciation of some of the problems involved in the programming of applications which interact with the X window system. Includes studying event-driven programming techniques.

Booklist

Students are likely to need ready access to the following:

L. Ameraal. *C for Programmers*. Wiley, 1991.

C. Brown. *UNIX Distributed Programming*. Addison Wesley, 1994.

A. Kelley and I. Pohl. *A Book on C: programming in C.* Benjamin Cummings, 3rd edition, 1995.

N.B. Access to either Ameraal or Kelley and Pohl will normally be sufficient.

The following should be consulted for different approaches or for further information:

J. Cox and B. Goodheart. *The Magic garden*. Prentice Hall, 1994.

L.W. Cannon et.al. Indian Hill C Style and Coding Standards. Technical report, University of Toronto, 1990. This highly recommended document is available in the home directory of user cmk as misc/c/docs/ihstyle.ps.

Andrew Koenig. *C - Traps and Pitfalls*. Addison Wesley, 1988.

B.W. Kernighan and D. Ritchie. *The C Programming Language - 2nd Edition*. Prentice Hall, 1988. 4.1

CS25010 Computer Communications

General Description

The purpose of this module is to present an introduction to the problems encountered and the methods used in modern computer based communications systems. The module is structured around the Open System Interconnection (OSI) model promulgated by the International Standards Organisation.

Commitment	Number of lectures	24
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
Other staff	Mr. Dave Price	
Pre-requisites	Pass or exemption in Computer Science at Level 1 or CS10310 by arrangement with the department	
Co-requisites	None	
Incompatibilities	None	
Assessment	Assessed coursework - 20% Written exam - 80%	
Timing	This module is offered only in Semester 1	

Aims

This module will present an introduction to the basis of the methods and technology used in modern computer based communications systems. It covers the basic concepts and terminology of point to point and broadcast communications over both public and private networks. Ethernet is studied in detail as an example of a Local Area network.

Objectives

At the end of the module, the students will understand

- the critical importance of standards
- the basic terminology and techniques of computer communications
- the way in which information may be coded
- some of the problems and features of communication protocols
- the components used to construct an Ethernet
- the common facilities available from network operators
- the functioning of a network file system
- the requirement for security

Syllabus

1. STANDARDS — 2 lectures
   The needs for standards and the areas they cover. The standards setting process and the standards setting bodies - including BSI, ISO, ANSI, IEEE, CCITT and IAB. Local and organisational standards and their relationship to national and international standards. The ISO Model. The weakness of standards and the standard setting process. De facto standards.

2. BASICS OF DATA COMMUNICATION — 4 lectures
   Analogue and digital data transmission; synchronous and asynchronous transmission; parallel and serial transmission. Modems, concentrators, time division and statistical multiplexors. Co-ax, twisted pair, fibre optic media. Speed, distance and error rates of the various transmission media.

3. INFORMATION AND CODING — 4 lectures
   Elements of information theory; formal definition of information content and redundancy. Fixed and variable length codes; optimal coding. Error detecting and error correcting codes; simple examples. The ASCII code and its characteristics.

4. COMMUNICATIONS PROTOCOLS — 5 lectures
   Datalink protocols. ISO; network, transport, FTAM. ARPA; IP, TCP, UDP, FTP, Telnet. X25 Protocols; PSE, PAD, SVC/PVC.

5. LOCAL AREA NETWORKS — 3 lectures
   Bus, ring and star topologies. Cost of attaching devices to networks. Standard protocols. Project 802.

6. WIDE AREA COMMUNICATIONS AND NETWORKS — 3 lectures
   The idea of a common carrier. The UK carriers; British Telecom, Mercury, Kingston and the new market entrants. Public switched networks and private lines. Kilostream, Megastream, Satstream, Videostream and similar services. ISDN. X25. Examples of wide area networks. Telecommunications history.

7. DISTRIBUTED COMPUTING — 2 lectures
   Remote file and disc access. Distributed processing. SUN NFS and similar systems.

8. SECURITY OF INFORMATION SYSTEMS — 1 lecture
   The need for security and its cost; risk assessment. Simple techniques such as passwords and badge readers; deficiencies of simple approaches. The Orange Book standard.

Booklist

It is considered essential to purchase the following:

William Stallings. *Data and Computer Communications*. 4th edition.

The following should be consulted for different approaches or for further information:

R. Deasington. *X25 Explained*. Ellis Horwood, 2nd. edition, 1987.

W. Stallings. *ISDN and Broadband ISDN*. Maxwell McMillan International, 2nd. edition, 1992. 4.1

CS25410 Digital System Design

General Description

The design of digital systems, at one time the sole domain of physicists and electrical engineers is now of great importance to computer scientists. This module provides a semi-formal approach to the design of digital systems and introduces CAD tools and techniques that support the design process. The module will survey a range of technologies with an emphasis on field-programmable logic. Emphasis will be placed on the software engineering aspects of the design tools and the parallels that can be drawn between the design of software systems and the hardware design techniques used.

Commitment	Number of lectures	14
	Number of workshops	0
	Number of practicals	10 x 2 hours
Co-ordinator	Dr. Fred Long	
Other staff	Not yet known	
Pre-requisites	**CS15010**	
Co-requisites	**CS21020**/C210	
Incompatibilities	None	
Assessment	Assessed coursework - 100%	
Timing	This module is offered only in Semester 1	

Aims

To give an understanding of the problem area and develop a systematic approach to the solution of a system's architecture and control.

- Introduction to the basic building blocks.
- An appreciation of design approach and methodology.
- Familiarisation of a range of current FPD's including: PAL's, PLA's and FPGA's.
- Familiarisation of CAD tools.
- Practical considerations and limitations of PLD's.
- Experience in the practical implementation of solutions using field programmable logic devices and computer-based design tools.

Objectives

To enable the design and implementation of complex logic circuits using software tools in design, realisation and testing.

- A thorough understanding of sequential logic circuits including counters, shift-registers and multiplexers.
- Use of top-down design approach and modularity in the hardware domain.
- A detailed experience of the use of specific devices in a number of applications.
- An appreciation of an integrated CAD environment using an industrial standard CAD tool.
- A detailed working knowledge of the PLD language and environment in the software design, software implementation and software testing of PLD's.
- An awareness of State Machine pitfalls.
- Appropriate application of PAL's, PLD's and FPGA's.

Syllabus

1. INTRODUCTION — 1 lecture
   Course organisation; Recap of logic; What are PLD's; Design environment and language.
2. BUILDING BLOCKS — 2 lectures
   Sequential logic devices; Counters; Shift-registers; Multiplexers.
3. DESIGN ENVIRONMENT — 3 lectures
   PLD language; Indexed equations; Tables; Streams; Testing.
4. DESIGN METHODS — 4 lectures
   Top-down; Control and architecture; Algorithmic State Machines; Notation; Transition tables; Realisation.
5. DESIGN ENVIRONMENT — 2 lectures
   PLD and ASM's; Procedures; Integrated environment.
6. PRACTICALITIES — 2 lectures
   Design pitfalls; Fixed product terms; Fixed I/O pins; Registers; Propagation times.
7. LABORATORY WORK — 10 practicals
   This section of the course will be concerned with giving the student experience of digital system design using the methods introduced in lectures. A number of design problems, of increasing complexity will be set. The student will obtain experience of the process of moving from a well specified problem, through a systematic design procedure, to a realisation in software and hardware. All practical work will require documentation as discussed.

Booklist

It is considered essential to purchase the following:

Franklin P. Prosser and David E. Winkel. *The Art of Digital Design*. Prentice-Hall International Editions, 2nd edition, 1987.

Students are likely to need ready access to the following:

Randy H. Katz. *Contemporary Logic Design*. Benjamin/Cummings Publishing, 1st edition, 1994.

Neil Storey. *Electronics: A Systems Approach*. Addison Wesley, 1st edition, 1992.

The following should be consulted for different approaches or for further information:

Martin S. Roden. *Micro-Cap IV*. Benjamin/Cummings Publishing, 1st edition, 1993. 4.1

CS26210 Artificial Intelligence Concepts

General Description

Artificial Intelligence (AI) has made many important contributions to computer science in general, and most experts believe AI techniques will become increasingly important. This module introduces students to the fundamental concepts of AI. Key issues including knowledge representation and reasoning, expert systems and learning are addressed both theoretically and practically.

Commitment	Number of lectures	12
	Number of workshops	12
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
Other staff	Not yet known	
Pre-requisites	Pass or exemption in Computer Science at Part I or CS10310 by arrangement with the department	
Co-requisites	None	
Incompatibilities	**CS36210**	
Assessment	Assessed coursework - 40% Written exam - 60%	
Timing	This module is offered only in Semester 2	

Aims

Artificial Intelligence is the study of computer systems which can perform the sort of tasks that are usually associated with human intelligence. Examples are: chess playing, pattern recognition, speech understanding and problem solving. The aim of this module is to introduce the main ideas and current problems in Artificial Intelligence including the key concepts of knowledge representation and reasoning, expert systems and learning. Students will be required to implement these concepts using an Artificial Intelligence programming language.

Objectives

On successful completion of this module students should:

- understand the importance of knowledge representation and their role in Artificial Intelligence systems;
- understand the use search in the solution of problems such as path planning and game playing;
- be familiar with the principle techniques used in implementing machine learning;
- understand the fundamentals of first generation expert system technology, and the conceptual basis of current attempts to overcome its limitations;
- be capable of implementing fundamental Artificial Intelligence algorithms in the Artificial Intelligence language Lisp.

Syllabus

1. INTRODUCTION — 1 lecture
   The origins of Artificial Intelligence; 4 definitions of Artificial Intelligence. Characterising good and bad problem domains.
2. SEARCH — 2 lectures
   The role of search in Artificial Intelligence. Search strategies: basic search; heuristic search; and game playing. Two workshops implementing depth- and breadth-first search, heuristic search and search space pruning.

3. KNOWLEDGE REPRESENTATION AND INFERENCE — 3 lectures
   Issues in knowledge representation: representation adequacy; inferential adequacy; inferential efficiency; acquisitional efficiency. Knowledge Representation formalisms: propositional and predicate logic, procedural representations; semantic nets; frame-based representations.

4. GENETIC ALGORITHMS — 1 lecture
   Chromosomes and hereditary traits; survival of the fittest; crossover; the rank method.

5. EXPERT SYSTEMS — 2 lectures
   Characterising first generation expert systems. Expert system structure: knowledge representation; control strategies. Limitations of current expert system technology.

6. LEARNING — 3 lectures
   The role of learning in Artificial Intelligence. Symbollic approaches to learning: rote learning; Winston's arch learning program; version spaces and CYC.

7. LISP — 12 practicals
   Workshops on programming in Lisp.

Booklist

It is considered essential to purchase the following:

E. Rich and K. Knight. *Artificial Intelligence*. McGraw Hill, 2nd. edition, 1991.

Students are likely to need ready access to the following:

Charniak and McDermott. *Introduction to Artificial Intelligence*. Addison Wesley, 1985.

Robert Wilensky. *Common LISPcraft*. W.W. Norton, New York, 1986.

P. H. Winston. *Artificial Intelligence*. Addison Wesley, 3rd. edition, 1992.

The following should be consulted for different approaches or for further information:

S. C. Shapiro. *Encyclopedia of Artificial Intelligence*. Addison-Wesley, 1992. 4.1

CS27310 Relational Database Systems

General Description

This module introduces fundamental principles of database design and implementation. It covers practical topics concerned with entity-relationship modelling and effective use of the facilities provided by Oracle and theorectical topics concerned with data modelling, placing particular emphasis on the relational data model, relational algebra, relational calculus, and the realisation of the relational model in Oracle.

Commitment	Number of lectures	24
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
Other staff	Not yet known	
Pre-requisites	Pass or exemption in Computer Science at Part I or CS10310 by arrangement with the department	
Co-requisites	None	
Incompatibilities	None	
Assessment	Assessed coursework - 40% Written exam - 60%	
Timing	This module is offered only in Semester 1	

Aims

This module aims to familiarise students with the techniques used in designing and implementing database systems, and with the concepts embodied in relational database systems.

Objectives

On successful completion of this module, students will be able to:

- design and validate a database from a given enterprise description;
- implement a database design using Oracle;
- describe and use the facilities of a modern database system;
- explain and provide a rationale for database concepts;
- understand and use a data model, and in particular, the relational data model;
- improve and extend a database design.

Syllabus

1. DATABASE SYSTEMS CONCEPTS — 1 lecture

2. RELATIONAL MODELLING I — 6 lectures
   Entities and relationships. Connection traps. The design of relations. Transformation of E-R model into Relational Schema.
3. IMPLEMENTING A DATABASE USING ORACLE — 3 lectures

4. RELATIONAL MODELLING II — 3 lectures
   Data normalisation. Validating a design.

5. ADVANCED SQL — 4 lectures
   Views and Indexes. Integrity constraints in Oracle. Built-in functions. Nested Queries and subqueries. Null values and the outer join. The role of the data dictionary. The data dictionary in ORACLE.

6. INTERFACES AND TOOLS — 1 lecture
   The Forms environment SQL in applications programs The Report Writer

7. RELATIONAL DATABASE DESIGN METHODOLOGY — 4 lectures
   An overview of design techniques. Overview of the database design process. First level design. Second level design.

8. DATA MODELS AND ORACLE — 2 lectures

Booklist

Students are likely to need ready access to the following:

C.J. Date. *An Introduction to Database Systems*. Addison-Wesley, 6th edition, 1995.

J.G. Hughes. *Database Technology, A Software Engineering Approach*. Prentice-Hall, 1988.

Computer unit documentation. To be advised.

Rick F. van der Lans. *The SQL Guide to Oracle*. Addison-Wesley, 1991. 4.1

CS28010 Personal Communication Skills

General Description

This module covers both written and spoken communication skills. Although the topics are applicable more widely than simply to software engineering, they are taught in that context. The module is not therefore suitable for students from other disciplines. The module is an obligatory part of the BEng and MEng degree schemes in Software Engineering and of the BSc schemes in Computer Science and Computer Science with a European Language.

Commitment	Number of lectures	15
	Number of workshops	0
	Number of practicals	5 x 1 hour
Co-ordinator	Dr. Fred Long	
Other staff	Not yet known	
Pre-requisites	None	
Co-requisites	**CS21020**/C210	
Incompatibilities	None	
Assessment	Assessed coursework - 100%	
Timing	This module extends over both Semester 1 and Semester 2	

Aims

The aim of this module is to improve students' skills in oral and written communication.

Objectives

On successful completion of this module, students will be able to:

- select an appropriate typographical style and layout for a simple document;
- write a short technical document using clear, correct and simple English;
- avoid the commoner solecisms in written English;
- plan and deliver a short technical presentation, using appropriate visual aids;
- play an effective role in a more complex group presentation;
- plan and chair an effective meeting.

Syllabus

1. TYPOGRAPHICAL DESIGN — 4 lectures
   Fonts: sizes, styles and families. Paragraph styles and justification. Page layouts, headers and footers. Styles for headings. The purpose of typographical design: logical and physical document structures. The need for consistency: style sheets.
2. HOW TO WRITE GOOD ENGLISH — 6 lectures
   The need to write clear, concise, correct and intelligible technical English. Choice of words, use of apostrophes, punctuation. Common faults of style: verbosity, excessive complication, overuse of the passive, hackneyed phrases. Differences of register.
3. PLANNING A PRESENTATION — 4 lectures
   Styles of presentation. Matching a presentation to its audience. Structuring and planning a presentation. Use of visual aids. Delivery.

4. MEETINGS 1 lecture
Purpose of meetings. Conduct of meetings. Chairing a meeting.

Booklist

The following may be helpful:

G. Barlow and S. Eccles. *Typesetting and Composition*. Chapman and Hall, 1992. A book on printing and typography suitable for computing students,worth looking at in the library but not worth buying unless you are particularly interested in the topic.

The Economist Style Guide. Economist Books Ltd., 1991. This book was originally produced as a source of guidance for contributors to the magazine The Economist, which is well known for the clarity and directness of its writing. Its coverage is much narrower than that of Weiner or Gowers, but it has much useful material in it.

Anne Eisenberg. *Effective Technical Communication*. McGraw-Hill, New York, 2nd. edition, 1992. This book has a wide coverage. It includes chapters on letters, abstracts, proposal writing, technical presentations, the use of graphics , and so on. There is much that is useful in the book but it is superficial in places.

Sir Ernest Gowers. *The Complete Plain Words*. HMSO, London, 3rd. edition, 1986. Revised by Sidney Greenbaum and Janet Whitcut. The author was a senior civil servant; he wrote this book in an attempt to improve the writing of his colleagues. The level is quite advanced and it presupposes a working knowledge of grammatical terminology; nevertheless, it is quite easy to read.

W. Strunk Jr. and E.B. White. *The Elements of Style*. Macmillan, 3rd. (revised) edition, 1982. This is a short book with particularly good coverage of common failings in grammar and punctuation. It is American and so, in a few places, the usage is not what you would expect in English. The emphasis of the book is more literary than technical. Despite these features, the book is recommended; if you take notice of everything it says, you will learn to write well.

J. Kirkman. *Good Style: Writing for Science and Technology*. E and F N Spon (Chapman and Hall), 1992. Although it doesn't have the encyclopaedic coverage of Fowler or Gowers, this is probably the best single book in the field to buy. It is aimed directly at the clear and straightforward style that is most suitable for technical writing.

P. Race. *500 Tips for Students*. Blackwell, 1992. A book full of hints on most aspects of university life. Topics covered range from revision and examination techniques, through to preparation and delivery of presentations.

Joan van Emden and Jennifer Easteal. *Report Writing*. McGraw-Hill, Maidenhead, 1987. This book is a good general introduction to report writing, with emphasis on the structure and style of the report as a whole, rather than on detailed points of grammar and style. It is written specifically for students.

E.S.C. Weiner. *The Oxford Guide to English Usage*. Clarendon Press, Oxford, 1983. A good up to date guide to issues of grammar, spelling and punctuation. 4.1

CS32110 Interactive Computer Graphics

General Description

This course introduces students to the fundamental principles and practice of interactive computer graphics, with an emphasis on: computational models and display of objects; graphical modelling standards; and virtual reality environments.

Commitment	Number of lectures	24
	Number of workshops	0
	Number of practicals	6 x 1 hour
Co-ordinator	Dr. Fred Long	
Other staff	Not yet known	
Pre-requisites	**CS21020**/C210	
Co-requisites	**CS23710**/C237(h) or a basic knowledge of the C language	
Incompatibilities	None	
Assessment	Assessed coursework - 20%	
	Written exam - 80%	
Timing	This module is offered only in Semester 2	

Aims

This module intends to introduce to students the fundamental concepts of interactive computer graphics including:

- the underlying mathematics of 2- and 3-dimensional graphics display techniques;
- 3-dimensional geometric modelling systems;
- modelling standards;
- virtual reality environments.

Objectives

The course concentrates on 3-dimensional wireframe display techniques, and on techniques for displaying solid objects. Geometric transformations and animation are also studied. This module involves practical graphics programming and students will need a basic knowledge of the C programming language. On successful completion of this module students should:

- have an understanding of 2- and 3-dimensional computer graphics display techniques, and implications for effective use of graphics systems;
- have both a theoretical and a practical understanding of 3-dimensional geometric modelling systems;
- understand the principal graphics standards and in particular PHIGS (Programmer's Hierarchical Interactive Graphics System) standard;
- have built and animated hierarchical graphics models using PHIGS;
- understand the variety of hardware and software technologies that underpin virtual reality systems, allowing natural control in three-dimensional interactive environments.

Syllabus

1. GRAPHICS SYSTEMS OVERVIEW — 1 lecture
   Important definitions, major components and features of interactive graphics systems.

2. THE INTERACTIVE GRAPHICS PARADIGM — 2 lectures
   Basic concepts: display space, viewports, windows, world coordinates, normalised device coordinates, and device coordinates.
3. COORDINATE TRANSFORMS — 2 lectures and 1 practical
   Trigonometry and matrix algebra. Basic 2-d and 3-d transformations, matrix representation and homogeneous coordinates. Composite transforms.
4. THE 2-D VIEWING PIPELINE — 2 lectures
   2-d transformations from world coordinates to device coordinates. 2-d clipping.
5. 3-D DISPLAY SYSTEMS — 3 lectures
   3-d viewing parameters. Perspective and parallel projection. 3D clipping.
6. INTRODUCTION TO GEOMETRIC MODELLING — 2 lectures
   Object models in 2-d and 3-d. Wireframe modelling.
7. OVERVIEW OF THE MAJOR GRAPHICS STANDARDS — 2 lectures
   The need for standards; the main ISO graphics standards.
8. PROGRAMMERS' HIERARCHICAL INTERACTIVE GRAPHICS SYSTEM (PHIGS) — 4 lectures and 3 practicals
   PHIGS programming model, output primitives and attributes, model creation and editing, display, graphical input. Multiple views and animations. Constructing hierarchical models; viewing models; animating models.
9. VIRTUAL REALITY (VR) AND ITS APPLICATIONS. — 3 lectures and 1 practical
   This section of the course will survey the theory and development of virtual reality systems. These techniques encompass a variety of hardware and software technologies and allow natural control in three-dimensional interactive environments. Lectures will cover interface technology for VR, human factors of VR systems, performance issues of VR systems, architectures, applications and case studies. Building virtual worlds; interacting with virtual worlds.
10. SOLID MODELLING AND VISIBLE SURFACE DETERMINATION — 3 lectures and 1 practical

Booklist

Students are likely to need ready access to the following:

J. D. Foley, A. van Dam, S. K. Feiner, J. F. Hughes, and R. L. Phillips. *Introduction to Computer Graphics*. Addison-Wesley, 1993.

A. Watt. *Fundamentals of Three-Dimensional Computer Graphics*. Addison-Wesley, 1989.

The following should be consulted for different approaches or for further information:

T. L. J. Howard, W. T. Hewitt, R. J. Hubbold, and K. M. Wyrwas. *A Practical Introduction to PHIGS and PHIGS PLUS*. Addison-Wesley, 1991. 4.1

CS33210 Advanced Software Engineering

General Description

This module introduces students to some advanced aspects of software engineering, specifically formal methods and software engineering environments.

Commitment	Number of lectures	24
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
Other staff	Not yet known	
Pre-requisites	**CS21020**/C210, **CS22110**/C220, **CS22210**/C220	
Co-requisites	None	
Incompatibilities	None	
Assessment	Assessed coursework - 20% Written exam - 80%	
Timing	This module is offered only in Semester 1	

Aims

This module aims to introduce students to two advanced aspects of software engineering: the use of formal methods in software specification (with an emphasis on the ideas involved, rather than the formality); and software engineering environments and tool support interfaces.

Objectives

On successful completion of this course students should understand:

- the problems of incompleteness, inconsistency and ambiguity arising from traditional methods of software specification;
- the different approaches to formal specification;
- the stages and processes involved in the development of a software design using VDM;
- the deficiencies of VDM and the attempts to overcome these in some other formal specification methods;
- the need for database support for project management and software development;
- the necessity for standard tool support interfaces;
- the basic facilities of the tool support interface PCTE;
- the technical problems facing tool support interface developers;
- the current trends in software tool integration.

Syllabus

1. THE TRADITIONAL APPROACH TO SPECIFICATION — 1 lecture
   Problems of incompleteness, inconsistency and ambiguity. Practical problems (volume of paperwork, etc.).
2. FORMAL SPECIFICATIONS — 1 lecture
   The advantages and disadvantages of formal specification. Algebraic and operational specifications.
3. VDM AS A SPECIFICATION LANGUAGE — 5 lectures
   Introduction and history. The VDM specification language. Data types in VDM. An example specification.

4. VDM AS A FORMAL DEVELOPMENT METHOD — 4 lectures
   Stages and processes in the development of a software design using VDM. Data reification and operation decomposition.

5. OUTSTANDING PROBLEMS AND OTHER METHODS — 1 lecture
   Formal specification of systems with concurrency. Modularisation of formal specifications. Safety and reliability issues. Other specification languages, Z and GYPSY (briefly). ANNA and SPARK.

6. INTEGRATION OF SOFTWARE TOOLS — 2 lectures
   Why integration is important. The dimensions of integration. Why traditional operating systems are not good enough.

7. INTEGRATED PROJECT SUPPORT ENVIRONMENTS — 4 lectures
   Database support for project management and software development. Typical tools. Use of a database as a means of integration. Other aspects of integration (e.g., the user interface). Structure of an IPSE; Tool Builder as an example.

8. TOOL SUPPORT INTERFACES — 4 lectures
   The need for standard tool support interfaces. The issues which such interfaces should address. Current standards (PCTE and CAIS). Technical problems (granularity, wide area distribution).

9. CURRENT TRENDS — 2 lectures
   A brief discussion of current trends in tool integration and CASE.

Booklist

It is considered essential to purchase the following:

C.B. Jones. *Systematic Software Development Using VDM*. International Series in Computer Science. Prentice-Hall, 2nd. edition, 1990.

The following should be consulted for different approaches or for further information:

A.W. Brown, D.J. Carney, E.J. Morris, D.B. Smith, and P.F. Zarrella. *Principles of CASE Tool Integration*. Oxford University Press, 1994.

I. Hayes, editor. *Specification Case Studies*. International Series in Computer Science. Prentice-Hall, 2nd. edition, 1993.

J.M. Spivey. *The Z Notation: A Reference Manual*. International Series in Computer Science. Prentice-Hall, 2nd. edition, 1992.

J. Woodcock and M. Loomes. *Software Engineering Mathematics*. Pitman, 1988. 4.1

CS33310 Formal Methods in Software Engineering

General Description

The course develops more rigorously the formal methods introduced in **CS33210**. As background to the formal methods in software engineering, students are introduced to formal logic and some of its other applications in Computer Science

Commitment	Number of lectures	24
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
Other staff	Not yet known	
Pre-requisites	**CS33210**	
Co-requisites	None	
Incompatibilities	None	
Assessment	Assessed coursework - 20% Written exam - 80%	
Timing	This module is offered only in Semester 2	

Aims

This module aims to develop more rigorously the formal methods introduced in **CS33210** by presenting mathematical logic in a more formal way, illustrating a number of different logics, and by showing some of the other applications of logic in Computer Science.

Objectives

On successful completion of this course students should understand:

- the nature of formal proof in the propositional and predicate logics;
- the need for the three valued logic of partial functions and how to carry out formal proof in that logic;
- the ideas behind correctness proofs of computer code and how to carry out such proofs themselves;
- how proof is applied to the data types of VDM;
- temporal logic and its application to the specification of concurrent systems;
- the applications of logic in resolution theorem proving, logic programming, expert systems, and fuzzy systems.

Syllabus

1. PROPOSITIONAL CALCULUS — 2 lectures
   Revision of formal logic. Truth tables. The logic of propositions. Logical reasoning. Consistency, completeness, and decidability.
2. PREDICATE CALCULUS — 4 lectures
   The language of predicates. The use of quantifiers. Valuations and models. Deductions, axioms and proofs.
3. CORRECTNESS PROOFS — 3 lectures
   Programming proof rules and their application in the specification of computer systems.

4. THE LOGIC OF PARTIAL FUNCTIONS 2 lectures
   Proof in the three valued logic of partial functions. Differences from the traditional two valued logic. Applications to formal methods, in particular VDM.
5. PROOF INVOLVING VDM DATA TYPES 5 lectures
   The VDM data types. Proof using the data types.
6. TEMPORAL LOGIC 2 lectures
   Specification of time dependent systems. The operators **always**, **sometimes**, **next** and **until**. Extension of VDM to include temporal logic.
7. LOGIC APPLICATIONS IN COMPUTING 4 lectures
   Resolution theorem proving. Logic programming and Prolog. Multi-valued logics, Fuzzy logic, and reasoning with uncertainty.

Booklist

It is considered essential to purchase the following:

R.D. Dowsing, V.J. Rayward-Smith, and C.D. Walter. *A First Course in Formal Logic and its Application in Computer Science,*. Computer Science Texts. Blackwell Scientific Publications, 1986.

C.B. Jones. *Systematic Software Development Using VDM*. International Series in Computer Science. Prentice-Hall, 2nd. edition, 1990.

The following should be consulted for different approaches or for further information:

D. Bjørner, C.A.R Hoare, and H. Langmaack, editors. *VDM '90: VDM and Z - Formal Methods in Software Development*, volume 428 of *LNCS*. Springer-Verlag, 1990.

J.M. Spivey. *The Z Notation: A Reference Manual*. International Series in Computer Science. Prentice-Hall, 2nd. edition, 1992.

J. Woodcock and M. Loomes. *Software Engineering Mathematics*. Pitman, 1988. 4.2

CS34210 Topics in Programming Languages

General Description

This module addresses the basic ideas underlying compilers and other language processing systems. It then provides an opportunity for studying the formal description of programming language semantics.

Commitment	Number of lectures	24
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
Other staff	Not yet known	
Pre-requisites	**CS21020**/C210	
Co-requisites	None	
Incompatibilities	None	
Assessment	Assessed coursework - 20% Written exam - 80%	
Timing	This module is offered only in Semester 1	

Aims

This module aims to familiarise students with the concepts required for specifying and implementing programming languages.

Objectives

On successful completion of this course students should be able to:

- make effective use of compilers and other language processing software;
- apply the techniques and algorithms used in compilation to other areas of software engineering;
- understand the need for implementation independent semantics of programming languages;
- provide semantic interpretation for simple programming languages.

Syllabus

1. OVERVIEW OF COMPILATION SYSTEMS — 10 lectures
   The structure of a typical compiler and the functions of a linker. Formal specification of the lexis and syntax of a programming language. Lexical and syntactic analysis of a language and tools for generating analysers. The idea of an intermediate language. Code generation. Local and global optimisation. Structure of assemblers and linkers.
2. PROGRAMMING LANGUAGE SEMANTICS — 12 lectures
   Concepts of abstract syntax and denotational semantics for computer languages. Syntactic and semantic domains. The concept of state and evaluation functions. Summary of the lambda calculus notation. The semantics of a simple block structured language. Side effects and procedures. Continuation semantics. Standard semantics. The use of ML to express the functionality of a denotational semantics. The construction of an interpreter, in ML, of a simple block structured language.

Booklist

It is considered essential to purchase the following:

Lloyd Allison. *A Practical Introduction to Denotational Semantics*. Cambridge, 1986.

Students are likely to need ready access to the following:

A.V. Aho, R. Sethi, and J.D. Ullman. *Compilers: Principles, Techniques and Tools*. Addison-Wesley, 1986.

The following should be consulted for different approaches or for further information:

M.J.C. Gordon. *The Denotational Description of Programming Languages, An Introduction*. Springer-Verlag, 1979.

A. A. Wikström. *Functional Programming Using Standard ML.* Prentice-Hall, 1987. 4.1

CS35010 Telematics

General Description

Telematic systems are systems which combine information technology, telecommunications and possibly broadcasting. This module is designed to to provide an insight into the nature and potential of Telematic Applications together with an understanding of the component technologies.

Commitment	Number of lectures	24
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
Other staff	Not yet known	
Pre-requisites	**CS25010**/C250(h)	
Co-requisites	**CS23710**/C237(h)	
Incompatibilities	None	
Assessment	Assessed coursework - 20% Written exam - 80%	
Timing	This module is offered only in Semester 2	

Aims

This module is designed to provide an insight into the nature and potential of Telematics Applications together with an understanding of the component technologies. The module will also present the characteristics of new high speed technologies. As well as traditional lectures, some material will be presented by videos made by world experts in various fields.

Objectives

At the end of the module, the students will:

- understand the potential of telematics systems;
- be able to create a simple client-server pair of programs using TCP/IP network support;
- understand some of the problems that appear in the management of routing and naming in large networks;
- understand the issues involved in the management of networks.

Syllabus

1. TELEMATIC SYSTEMS — 6 lectures
   Remote Windowing Systems; Messaging; Information Access; Video/Audio/FileTransfer; MultiFunctional Workstations.
2. INTERNETWORKING — 7 lectures
   TCP/IP Protocols; TCP/IP Programming; Naming in Large Networks; Routing in Large Networks.
3. INTEGRATED SERVICES NETWORKS — 7 lectures
   N-ISDN; B-ISDN; Fast LANs; MANs; Optical Networking.
4. NETWORK PLANNING AND MANAGEMENT — 4 lectures
   Security; Queueing Theory; Network Management; Network Planning.

Booklist

Students are likely to need ready access to the following:

W. Stallings. *ISDN and Broadband ISDN.* Maxwell McMillan International, 2nd. edition, 1992.

The following should be consulted for different approaches or for further information:

W. Stallings. *Local and Metropolitan Area Networks*. Maxwell McMillan International, 4th. edition, 1993.

W. Richard Stevens. *TCP/IP Illustrated*, volume 1: The Protocols. Addison-Wesley, 1994. 4.1

CS35410 High Level Digital Design

General Description

This module builds upon the foundations of **CS15010** and **CS25410**, at the same time bringing together many other aspects of the course. Students will be introduced to VHDL - an IEEE language standard which is of increasing importance in the area of design, simulation and implementation of digital systems and which has strong links with Ada. In addition students will consolidate their experience of using a fully integrated development environment. The course will concentrate on the principles involved in micro-processor architectures and will involve students in the design, implementation, simulation and physical build of a simple micro-processor chip based on programmable logic.

Commitment	Number of lectures	14
	Number of workshops	0
	Number of practicals	10 x 2hrs
Co-ordinator	Dr. Fred Long	
Other staff	Not yet known	
Pre-requisites	**CS25410**	
Co-requisites	None	
Incompatibilities	None	
Assessment	Assessed coursework 100%	
Timing	This module is offered only in Semester 2	

Aims

The aims of this module are to:

- introduce high level design and simulation techniques for hardware;
- give practical experience of using these techniques;
- give an appreciation of the capabilities and shortcomings of current technology;
- enhance students' understanding of processor architectures;
- consolidate the knowledge gained in CS25410.

Objectives

On successful completion of this module students should be able to:

- specify logic designs in VHDL;
- use an integrated design environment to implement and simulate their designs;
- design, simulate and implement a simple microprocessor using a field programable device;
- test and validate the resultant devices by using a suitable test harness.

Syllabus

1. INTRODUCTION — 1 lecture
   Course overview.
2. THE DESIGN ENVIRONMENT — 1 lecture

3. VHDL 8 lectures
The VHDL language, with implementation examples.
4. MICROPROCESSOR ARCHITECTURE AND OPERATION 3 lectures
5. TRENDS 1 lecture
Current trends in logic devices and design methods.
6. LABORATORY WORK 10 practicals
This section of the course will be concerned with giving the student experience of high level digital design using the methods introduced in lectures. A number of design problems, of increasing complexity will be set. The practical work will culminate in the student designing, simulating and implementing a simple microprocessor, which must meet a pre-determined specification. All practical work will require documentation as discussed.

Booklist

Not appropriate. 4.1

CS36210 Artificial Intelligence Concepts

General Description

Artificial Intelligence (AI) has made many important contributions to computer science and most experts believe AI techniques will become increasingly important. This module introduces students to the fundamental concepts of AI. Key issues including knowledge representation, reasoning and learning are addressed both theoretically and practically.

Commitment	Number of lectures	24
	Number of workshops	0
	Number of practicals	6 x 1-hour
Co-ordinator	Dr. Fred Long	
Other staff	Not yet known	
Pre-requisites	**CS21020**/C210	
Co-requisites	None	
Incompatibilities	**CS26210**	
Assessment	Assessed coursework - 20% Written exam - 80%	
Timing	This module is offered only in Semester 1	

Aims

Artificial Intelligence is the study of computer systems which can perform the sort of tasks that are usually associated with human intelligence. Examples are: chess playing, pattern recognition, speech understanding and problem solving. The aim of this module is to introduce the main ideas and current problems in Artificial Intelligence including the key concepts of knowledge representation and reasoning, expert systems and learning. Students will be required to implement these concepts using an Artificial Intelligence programming language.

Objectives

On successful completion of this module students should:

- understand the importance of knowledge representation and their role in Artificial Intelligence systems;
- understand the use search in the solution of problems such as path planning and game playing;
- be familiar with the principle techniques used in implementing machine learning;
- understand the fundamentals of first generation expert system technology, and the conceptual basis of current attempts to overcome its limitations;
- recognise the distinction between symbolic approaches to Artificial Intelligence and sub-symbolic approaches;
- be capable of implementing fundamental Artificial Intelligence algorithms in the high level Artificial Intelligence language POP-11.

Syllabus

1. INTRODUCTION — 1 lecture and 6 practicals
   The origins of Artificial Intelligence; 4 definitions of Artificial Intelligence. Characterising good and bad problem domains.
2. INTRODUCTION TO POP-11 — 3 lectures
   Poplog environment, TEACH, HELP, basic POP-11, list processing, recursion, advanced features.

3. Knowledge Representation — 3 lectures
   Issues in knowledge representation: representation adequacy; inferential adequacy; inferential efficiency; acquisitional efficiency. Knowledge Representation formalisms: propositional and predicate logic, procedural representations; semantic nets; frame-based representations.

4. Search — 2 lectures
   The role of search in Artificial Intelligence. Search strategies: basic search; heuristic search; and game playing. Two workshops implementing depth- and breadth-first search, heuristic search and search space pruning.

5. Expert Systems — 2 lectures
   Characterising first generation expert systems. Expert system structure: knowledge representation; control strategies. Limitations of current expert system technology.

6. Learning — 3 lectures
   The role of learning in Artificial Intelligence. Symbollic approaches to learning: rote learning; Winston's arch learning program; version spaces and CYC.

7. Qualitative and Model-Based Reasoning — 4 lectures
   Techniques and methods used to provide second generation expert systems. Qualitative vs. quantitative representation and reasoning. Modelling physical domains and naive physics. Structure, behaviour, function and purpose. Models for simulation and diagnosis.

8. The Physical Symbol Hypothesis — 1 lecture
   Strong and weak Artificial Intelligence - philosophical interlude.

9. Neural Nets — 2 lectures
   Simulated neural networks. Hill climbing and back propagation. Characteristics of back propagation.

10. Genetic Algorithms — 1 lecture
    Chromosomes and hereditary traits; survival of the fittest; crossover; the rank method.

Booklist

It is considered essential to purchase the following:

E. Rich and K. Knight. *Artificial Intelligence*. McGraw Hill, 2nd. edition, 1991.

Students are likely to need ready access to the following:

Charniak and McDermott. *Introduction to Artificial Intelligence*. Addison Wesley, 1985.

Ramsay, Barrett, and Sloman. *POP-11: A Practical Language for Artificial Intelligence*. Ellis Horwood, 1985.

P. H. Winston. *Artificial Intelligence*. Addison Wesley, 3rd. edition, 1992.

The following should be consulted for different approaches or for further information:

S. C. Shapiro. *Encyclopedia of Artificial Intelligence*. Addison-Wesley, 1992. 4.1

CS36310 Artificial Intelligence Systems

General Description

This module concentrates on the technology and methodology of Artificial Intelligence practice, in particular, programming languages and software systems. The aim is to build on the concepts introduced in **CS36210** and show how knowledge-based systems can be implemented, developed and evaluated.

Commitment	Number of lectures	12
	Number of workshops	6
	Number of practicals	6 x 1-hour
Co-ordinator	Dr. Fred Long	
Other staff	Not yet known	
Pre-requisites	**CS36210**	
Co-requisites	None	
Incompatibilities	None	
Assessment	Assessed coursework - 20%	
	Written exam - 80%	
Timing	This module is offered only in Semester 2	

Aims

This course will provide a good understanding of a variety of AI systems from machine learning through adaptive computing to expert systems. This will enable students to gain an understanding of the current state of the art AI systems.

The emphasis on the course will be in not only practical applications but also on practical experience of the A.I. systems concerned. It will also emphasize issues associated with the construction of such systems (i.e. problems in building systems for real applications as well as issues of ‘software’ engineering AI systems).

The main subject areas covered will include Evolutionary Computing (GAs and Classifier systems), Neural Networks, Machine Induction, Case Based Reasoning, Expert Systems and Model-Based Reasoning.

The structure of this module is that during the 20 lecture series a number of workshop sessions will run instead of the standard lectures.

Note: students will be expected to undertake a significant amount of self-study throughout this course.

Objectives

On successful completion of this module students should:

- be aware of the implications of a problem domain for the selection of knowledge acquisition methodology, knowledge engineering technique, software development methodology, and system evaluation strategy;
- understand the differences between the main classes of application oriented intelligent system, including representation and reasoning facilitites, and performance issues;
- understand the different techniques and methodologies available for knowledge acquisition, and their relative strengths and weaknesses;
- appreciate the problem of software engineering for intelligent systems;
- have had practical exposure to a range of intelligent systems: including an expert system shell; a CBR toolkit; a neural network simulator and a knowledge acquisition environment.

Syllabus

1. OVERVIEW OF AI SYSTEMS 1 lecture
   This lecture will cover the reasons for building Intelligent Systems, the problems associated with building such systems and the benefits companies have obtained from these systems.
2. CLASSES OF KNOWLEDGE BASED SYSTEM 1 lecture
   This lecture will introduce the various classes of intelligent system which can be constructed. It will introduce classification and construction tasks and consider their features.
3. EXPERT SYSTEMS 2 lectures and 1 practical
   These lectures will build on the introduction to experts systems provided in other AI models. It will introduce the students to commercial expert system shells and wil provide hands on experience of at least one system. It will condider the issues associated with constructing expert systems and of rule-based representations.
4. KNOWLEDGE ACQUISITION 2 lectures and 1 practical
   These lectures will consider the issue of obtaining the knowledge to be used in an expert system. They will be based around the KADS formalism which will be briefly introduced. This formalism guides the expert system building through the process of task identification and knowledge acquisition.
5. EVOLUTIONARY COMPUTING 2 lectures and 2 practicals
   The aim of these lectures will be to introduce the basic algorithms and systems used in evolutionary computing, in particular genetic algorithms and classifier systems. The lectures will introduce the various issues in evolutionary computing (such as representation, evaluation, selection, mutation and crossover) as well as available toolkits. To accompany these lectures practical sessions will be run where students will develop their own genetic algorithm application.
6. NEURAL NETWORKS 2 lectures and 1 practical
   These lectures will introduce the concept behind Neural Networks and investigate some of the more important and useful neural network architectures. They will also aim to give an appreciation of the application of neural computing in industry and how to proceed when constructing such a system. Students will gain experience of the a neural network simulator.
7. CASE BASED REASONING 2 lectures and 1 practical
   These lectures will provide the student with an understanding of the techniques which comprise a case-based reasoning (CBR) system. They should also gain an understanding of their strengths and weaknesses as well as when to apply and when not to apply CBR. The lectures will consider the basic structure of a CBR system and the use of a case base, representation and indexing of cases, case retrieval and adaptation, and finally case repair.

Booklist

Students are likely to need ready access to the following:

R. Beale and T. Jackson. *Neural computing : an introductio*. Institute of Physics, 1990.

Lawrence Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, 1991.

Janet L. Kolodner. *Case-based reasoning*. Morgan Kaufmann, 1993.

G. Luger and W. Stubblefield. *Artificial Intelligence and the Design of Expert Systems*. Benjamin/Cummings, 1989.

C. Price. *Knowledge Engineering Toolkits*. Ellis Horwood, 1990. 4.1

CS36410 Industrial Robotics

General Description

This course introduces the important elements of robotics through particular emphasis on automatic flexible assembly. The emphasis of the material is to provide an understanding of the software aspects of flexible automated assembly, though set in the wider context of general robotics and other computer-based automation. The particular challenges of this application area are presented along with the techniques currently available to tackle them.

Commitment	Number of lectures	24
	Number of workshops	0
	Number of practicals	5 hours
Co-ordinator	Dr. Fred Long	
Other staff	Not yet known	
Pre-requisites	**CS21020**/C210	
Co-requisites	**CS36210**	
Incompatibilities	None	
Assessment	Assessed coursework - 20% Written exam - 80%	
Timing	This module is offered only in Semester 1	

Aims

This module presents:

- an overview of robotics for software engineers;
- the software issues in the discipline as examples and extensions of the concepts introduced in other modules;
- the problems characteristic of the discipline;
- the range of techniques and algorithms brought to bear on these problems.

Objectives

On successful completion of this module, students should

- have an understanding of basic robotic equipment;
- have an understanding of the basic functional requirements of a software system for robot task definition;
- be familiar with the specifically robotics facilities of a range of robot programming languages and environments;
- understand the range of planning problems in the field and have a knowledge of some representative techniques used to solve them;
- understand the challenge which robotics and the 'behaviours' model represent to traditional AI techniques.

Syllabus

1. AUTOMATED FLEXIBLE ASSEMBLY AND ITS CONTEXT — 1 lecture
   The nature of the assembly problem; constraints and pressures on it.

2. Robot Systems — 4 lectures
   Overview of the components of a robot work cell; manipulator geometry; kinematics; drives; control regimes; end effectors; compliance; sensing; guarded movement. Manipulators; degrees of motion and freedom; joint, cartesian and relative cartesian motion; dextrous work envelopes and singularitues. Gripper mechanisms.
3. Explicit Robot Programming — 4 lectures
   Levels of task description. Teach mode and explicit programming. the languages VAL and AML. An example implementation of a trivial task in an end effector level language. Consideration of the nature of the development process.
4. Offline Programming Assistance — 2 lectures
   Robot modelling; solid body systems; simulation.
5. Implicit Robot Programming — 4 lectures
   Task specification; path planning; grasp planning; sequencing; program synthesis. The AL, RAPT and AUTOPASS systems.
6. Sensing — 3 lectures
   Sensory modalities; smart sensors; software implications; the perception problem.
7. Case Study — 1 lecture
   Study of the **InFACT** machine as an example of a market driven flexible automatic assemble machine.
8. The error problem — 2 lectures
   The practical requirements for error handling.
9. Behaviour Based Robot Programming — 3 lectures
   Outline of the concept; consideration of its benefits; examination of its contribution.

Booklist

Students are likely to need ready access to the following:

P.J. McKerrow. *Introduction to Robotics*. Addison-Wesley, 1991. 4.1

CS37410 Information Systems Analysis and Design

General Description

This module addresses the aims and principles of information systems analysis and design. It introduces a variety of analysis and design techniques, and shows how these are combined in analysis and design methods - in particular, in SSADM Version IV. It goes on to show how these central ideas can be applied to the development of distributed and object oriented information systems.

Commitment	Number of lectures	24
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
Other staff	Not yet known	
Pre-requisites	**CS27310** or CS27220/C272	
Co-requisites	None	
Incompatibilities	None	
Assessment	Assessed coursework - 20% Written exam - 80%	
Timing	This module is offered only in Semester 2	

Aims

This module aims to familiarise students with analysis of information requirements and design of database systems to meet those requirements.

Objectives

On successful completion of this module, students will be able to

- discuss the aims and objectives of an information system in the context of a human activity system,
- use a variety of analysis and design techniques to document existing information systems, to propose alternative new systems, and to specify required information systems,
- tune a database design specification to favour particular information requirements,
- combine these techniques within overall analysis and design methods,
- adapt analysis and design techniques and methods to meet the special needs of distributed information systems and object oriented information systems.

Syllabus

1. INFORMATION SYSTEMS AND HUMAN ACTIVITY SYSTEMS — 1 lecture
   Information systems as an enabling technology for human endeavour. Concepts: information, system, purposive system, human activity system, information system. Information systems in the context of a human activity systems. Information and data. The value of information as a resource or a commodity.
2. DATAFLOW MODELLING — 3 lectures
   Modelling the current system. Transforming a physical dataflow model to a logical dataflow model. Domain of change. Modelling proposed new systems. Modelling the required system.
3. REVIEW OF ENTITY-RELATIONSHIP MODELLING TECHNIQUES — 1 lecture
   The entity-relationship model as a description of the a universe of discourse. The entity-relationship model as a description of the structure of a database.

4. Entity-event modelling — 3 lectures
   Events and their effects on entity instances. Entities affected by an event. Entity event diagrams. Effect correspondance diagrams.
5. Relating different models of an information system — 1 lecture
   Entities and datastores. Entities and events. Events, processes and dataflows.
6. Analysis and design methods — 2 lectures
   Analysis and design aims. Questions to be addressed. Major steps in analysis and design. Analysis and design techniques. Combining techniques within design methods. SSADM Version IV.
7. Tuning a data model — 3 lectures
   Restructuring operations. When and how to restructure. Estimating processing performance. Estimating space usage. Derivation analysis. Temporal normal form analysis. Field grouping analysis. Key analysis. A design tuning method. Principal considerations in structural design. Structural design and implementation design.
8. Distributed information systems — 3 lectures
   Requirements and problems. Transparency. Approaches to distributed data provision. Centralised, distributed and federated databases. Processing distributed queries and transactions.
9. Object oriented data modelling — 3 lectures
   Object oriented design principles. Data design principles. Object oriented data design. An object oriented data design method. Using an object oriented database.

Booklist

Students are likely to need ready access to the following:

Caroline Ashworth and Laurence Slater. *An Introduction to SSADM Version 4*. McGraw-Hill, 1992.

C.J. Date. *An Introduction to Database Systems*. Addison-Wesley, 5th. edition, 1990.

E. Downs, P. Clare, and J. Coe. *SSADM Version 4*. Prentice-Hall, 1992.

Malcolm Eva. *SSADM Version 4: A User's Guide*. McGraw-Hill, 1992.

J.G. Hughes. *Database Technology, A Software Engineering Approach*. Prentice-Hall, 1988.

Fred R. McFadden and Jeffrey A. Hoffer. *Database Management*. Benjamin/Cummings, 1985.

N.B. Only one of Eva, Downs *et. al.* or Ashworth and Slater will be required.

4.1

CS37510 Database Management Systems

General Description

This module introduces the major components of a database management system and the techniques used in implementing them. It addresses data storage devices and systems, storage and retrieval strategies and mechanisms, query processing algorithms, transactions and concurrency control mechanisms, and data structures and algorithms used in ensuring database reliability and recovery from various kinds of failure.

Commitment	Number of lectures	24
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
Other staff	Not yet known	
Pre-requisites	None	
Co-requisites	**CS27310** (or CS27220/C272 as a pre-requisite)	
Incompatibilities	None	
Assessment	Assessed coursework - 20% Written exam - 80%	
Timing	This module is offered only in Semester 1	

Aims

This module aims to familiarise students with the techniques used in implementing database management systems.

Objectives

On successful completion of this module, students will be able to:

- discuss the relative merits of a variety of external storage devices;
- describe the components of a database management system in terms of the services they provide;
- explain and apply query processing strategies and techniques;
- compare and contrast a variety of transaction processing and concurrency control techniques;
- discuss the aims, objectives and relative merits of a variety of reliability and recovery mechanisms.

Syllabus

1. EXTERNAL STORAGE DEVICES AND SYSTEMS — 2 lectures

2. DATABASE COMPONENTS — 1 lecture

3. STORAGE AND RETRIEVAL STRATEGIES AND MECHANISMS — 9 lectures
   Primary access methods; classification of access methods, hashed access file organisation, indexed files (sparse index), B+ trees, files with a dense index. Files with variable length records. Secondary indices; inverted file organisation, multilist files, partitioned hash functions, partial match retrieval with signature trees. Hashing techniques for expandable files.
4. QUERY PROCESSING — 3 lectures
   Different kinds of query; ad-hoc, embedded, update. Need for query optimisation. Optimisation strategies; algebraic strategies, implementation dependent strategies. Optimising relational expressions. Generating

query plans for the optimised expression and selecting the cheapest. Query optimisation in some sample database management systems. Query optimisation by tableau minimisation.

5. TRANSACTIONS AND CONCURRENCY CONTROL 3 lectures
The nature of transactions. Transaction integrity, atomicity, permanence, independence. Concurrency control. Serializability, schedules. Locking. Starvation and deadlock, Deadlock prevention, Deadlock detection and resolution, Timestamping.

6. RELIABILITY AND RECOVERY 2 lectures
Major types of failure; aborted transactions, incorrect data, system failure, database destruction. Recovery mechanisms; backup, audit trail, checkpoint facility. The recovery manager. Recovery and restart procedures; restore and rerun, rollback roll forward. Recovering from different kinds of failure.

Booklist

Students are likely to need ready access to the following:

C.J. Date. *An Introduction to Database Systems*. Addison-Wesley, 5th. edition, 1990.

J.G. Hughes. *Database Technology, A Software Engineering Approach*. Prentice-Hall, 1988. 4.1

CS38010 Professional Issues in Software Engineering

General Description

This module covers a range of non-technical issues - legal, financial, managerial and organisational - with which a professional software engineer should be familiar. Although the topics are applicable more widely than simply to software engineering, they are taught in that context. The module is not therefore suitable for students from other disciplines. The module is an obligatory part of the BEng and MEng degree schemes in Software Engineering and of the BSc schemes in Computer Science and Computer Science with a European Language.

Long thin module

Commitment	Number of lectures	24
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
Other staff	Not yet known	
Pre-requisites	None	
Co-requisites	**CS33210** or by arrangement with the department	
Incompatibilities	None	
Assessment	Assessed coursework - 100%	
Timing	This module extends over both Semester 1 and Semester 2	

Aims

The aims of this module are to familiarise students with relevant non-technical aspects of the environment in which they are likely to be working after they graduate. In particular, it aims to give them an appreciation of relevant topics in:

- company organisation;
- finance and accounting;
- intellectual property rights;
- health, safety and liability;
- human resource management.

Objectives

On successful completion of this module, students should understand:

- the structure of the engineering profession and the role of professional institutions;
- the concept of a limited company and its financial structure;
- management styles and structure;
- the basis of financial and management accounting;
- intellectual property rights as they affect the computer industry;
- relevant aspects of industrial relations;
- safety, health and environmental requirements.

Syllabus

1. ORGANISATIONS AND THEIR STRUCTURES — 3 lectures
   Limited companies, private and public; partnerships; sole traders. Special features of limited companies; responsibilities of directors. Functional organisation and product line organisation. Centralised and decentralised organisations. Management styles. Management by objectives.
2. COMPANY FINANCE — 5 lectures
   The need for capital; investment and working capital; sources of funds; equity capital and loan capital. Cash flow and its importance. Costing: fixed costs and variable costs; overheads; opportunity costs; depreciation. Problems of cost allocation. Budgeting. Financial accounts: balance sheets, profit and loss accounts, sources and application of funds. Assessment of capital investment. Discounted cash flow analysis.
3. INTELLECTUAL PROPERTY RIGHTS — 4 lectures
   Use of patents, copyright and confidential information by software engineers in employment and across companies.
4. HEALTH AND SAFETY REQUIREMENTS — 5 lectures
5. INDUSTRIAL RELATIONS — 2 lectures
6. THE ENGINEERING PROFESSION — 1 lecture
   The structure of the engineering profession, both in the UK and abroad. Professional codes of conduct and codes of practice.

Booklist

Students are likely to need ready access to the following:

M.F. Bott, J.A. Coleman, J. Eaton, and D. Rowland. *Professional Issues in Software Engineering*. Pitman, 1991. 4.1

CS39030 Major Project

General Description

Students must complete a substantial piece of work, under the direction of a supervisor but involving self discipline, organisation and initiative. The projects typically involve developing a piece of software from an initial statement of requirements through specification and design stages, to successful implementation and testing. For students with a suitable background the project may contain a hardware element. A list of suggested projects is maintained by the third year co-ordinator. Projects are normally chosen from this list but we welcome students' own suggestions, especially those arising from industrial year experience.

Commitment	Number of lectures	4
	Number of workshops	0
	Number of practicals	0
Co-ordinator	Dr. Horst Holstein	
Other staff	Not yet known	
Pre-requisites	This course is for single honours Computer Science students, and for BEng students	
Co-requisites	None	
Incompatibilities	**CS39110**	
Assessment	Assessed coursework - 100%	
Timing	This module extends over both Semester 1 and Semester 2	

Aims

Single Honours Computer Science and BEng students are required to do a 30 credit major project as part of their undergraduate experience.

(Joint Honours students are not normally encouraged to do a CS39030 project, as opposed to a **CS39110** project because this would leave them short of taught modules.)

The purpose of the CS39030 project is to give students the opportunity to bring together the practical skills and theoretical understanding that they have acquired, by developing a complete product from the stage of specifying requirements through to demonstrating that the product does indeed satisfy those requirements. This process involves producing appropriate documentation, including documentation of the design decisions made.

Objectives

The objectives of this course are for the student to complete a substantial piece of work, under the direction of a supervisor but demonstrating self discipline, organisation and initiative.

The projects typically involve developing a piece of software from an initial statement of requirements through specification and design stages, to successful implementation and testing, although some may include a hardware element (for students with a suitable background).

A list of suggested projects is maintained by the third year co-ordinator. Projects are normally chosen from this list but we welcome students' own suggestions, especially those arising from industrial year experience.

Syllabus

1. FORMAL LECTURES — 4 lectures
   There are a small number of formal lectures associated with the project.

2. WRITTEN MATERIAL
   Various written material giving guidance on carrying out the project, assessment issues, and the project submission is provided by the department. Students are expected to spend perhaps ten hours in an average week on the project and to see their supervisor once a week. This course is assessed on the written report submitted by the student. Reports must be submitted by the end of the first week in May. Students will also be required to give a presentation of their project in the form of a seminar, which will account for 10% of the assessment.

3. ASSESSMENT
   A mark of at least 40% in CS39030 is required for an honours degree.

Booklist

Not appropriate. 4.1

CS39110 Minor Project

General Description

This course is intended for joint honours students who wish to undertake project work as part of their final year course. All details are like **CS39030**, except that the effort expected is less. It has proved possible in the past for joint honours students to undertake a project carrying more credit by finding a topic which includes elements from both computing and their other discipline; the resulting mark is then split appropriately between the two components of their degree. The Computer Science department encourages joint honours students to undertake such projects.

Commitment	Number of lectures	4
	Number of workshops	0
	Number of practicals	0
Co-ordinator	Dr. Horst Holstein	
Other staff	Not yet known	
Pre-requisites	This course is for joint honours Computer Science students	
Co-requisites	None	
Incompatibilities	**CS39030**	
Assessment	Assessed coursework - 100%	
Timing	This module extends over both Semester 1 and Semester 2	

Aims

This module is intended for joint honours students who wish to undertake project work as part of their final year course. All details are like CS39030, except that the amount of effort expected is only about three hours a week.

It has proved possible in the past for joint honours students to undertake a project equivalent to more than 10 credits by finding a topic which includes elements from both computing and their other discipline; the resulting mark is then split between the two components of their degree, counting for the equivalent of 10 credits in each component. The Computer Science department encourages joint honours students to undertake such projects.

Objectives

The objectives of this course are for the student to complete a substantial piece of work, under the direction of a supervisor but involving self discipline, organisation and initiative.

The projects typically involve developing a piece of software from an initial statement of requirements through specification and design stages, to successful implementation and testing, although some may include a hardware element (for students with a suitable background).

A list of suggested projects is maintained by the third year co-ordinator. Projects are normally chosen from this list but we welcome students' own suggestions, especially those arising from the other component of the student's degree scheme.

Syllabus

1. FORMAL LECTURES — 4 lectures
   There are a small number of formal lectures associated with the project.
2. WRITTEN MATERIAL
   Various written material giving guidance on carrying out the project, assessment issues, and the project submission is provided by the department. Students are expected to spend perhaps three hours in an average week on the project and to see their supervisor regularly. This course is assessed on the written report

submitted by the student. Reports must be submitted by the end of the first week in May. Students will also be required to give a presentation of their project in the form of a seminar, which will account for 10% of the assessment.

Booklist

Not appropriate. 4.1

CS41010 Object-Oriented Design

General Description

The major development in software design in the last decade has been the adoption of object-oriented techniques. The object approach first made an impact in the area of programming language design but the ideas have rapidly spread and object-oriented techniques are now of growing importance in areas as diverse as software design methods, databases, graphical user interfaces and distributed systems. This module provides an introduction to the key concepts and features of object technology. Students will learn object-oriented analysis and design and will implement object-oriented programs in Visualworks, a modern programming environment. The course will also provide a brief introduction to recent developments such as object request brokers.

Commitment	Number of lectures	20
	Number of workshops	6
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
Other staff	Not yet known	
Pre-requisites	C210/**CS21020** or equivalent experience	
Co-requisites	None	
Incompatibilities	Only available to students registered for Masters in Software Engineering	
Assessment	Assessed coursework - 50% Written Exam - 50%	
Timing	This module is offered only in Semester 1	

Aims

The aim of this course is to provide an outline of the key features of object-oriented technology. The course has a practical rather than theoretical emphasis: students will learn object-oriented design techniques by building object-oriented programs using a modern object-oriented programming language. As well as lectures, the course will use some small case studies and a significant design and implementation assignment.

Objectives

On successful completion of this module, students should:

- Understand the basic concepts of object technology;
- Have an appreciation of object-oriented analysis and design and methods;
- Have a working knowledge of an object-oriented programming language;
- Be able to customise an application framework to generate an implementation;
- Be able to produce simple design patterns.

Syllabus

1. INTRODUCTION — 4 lectures
   Motivation and background. History. From types to classes. Problems with ADTs. Programming by difference. Introduction to an object-oriented programming language.
2. OBJECT-ORIENTED ANALYSIS — 3 lectures
   Identifying objects in the problem domain. Identifying candidate classes. A look at some simple examples.

3. OBJECT-ORIENTED DESIGN — 3 lectures
   The design process.
4. A NOTATION FOR OOD. — 2 lectures
   Short case study using a suitable design process and notation.
5. REUSE OF DESIGNS — 2 lectures
   Using frameworks. Project: Customising a framework (e.g. Interviews, HotDraw).
6. DESIGNING FOR REUSE — 2 lectures
   Using design patterns to express important design concepts.
7. FURTHER ISSUES — 4 lectures
   A brief overview of Client-server systems, object-oriented databases, object request brokers.

Booklist

The following should be consulted for different approaches or for further information:

Grady Booch. *Object-Oriented Analysis and Design*. Addison-Wesley, 1994.

Timothy Budd. *An Introduction to Object-Oriented Programming*. Addison-Wesley, 1991.

James Rumbaugh et al. *Object-Oriented Modeling and Design*. Prentice-Hall, 1991.

Ian Graham. *Object-Oriented Methods*. Addison-Wesley, 2nd. edition, 1994.

W.R. Lalonde and J.R. Pugh. *Inside Smalltalk*, volume I. Prentice Hall, 1990.

Rebecca Wirfs-Brock, Brian Wilkerson, and Lauren Wiener. *Designing Object-Oriented Software*. Prentice Hall, 1990. 4.1

CS43010 Safety-related Systems

General Description

Modern industrial systems frequently have possible dangers, for workers, passengers, the general public and the environment. When they use computers to control hazardous actions, the software must be of particularly high quality, and independently assured for the safety-related purpose. The module deals with the standards concerned, and the technical implications for software engineers. Students are prepared for advanced study to take responsibility for the development of such systems.

Commitment	Number of lectures	24
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
Other staff	Not yet known	
Pre-requisites	C220/**CS22110**, C231(h)/**CS23110**	
Co-requisites	None	
Incompatibilities	Only available to students registered for Masters in Software Engineering	
Assessment	Assessed coursework - 20% Written exam - 80%	
Timing	This module is offered only in Semester 2	

Aims

To prepare students who may develop software-based systems for safety-related purposes. To introduce relevant national and international standards, and explain implications for software engineers.

Objectives

On successful completion of this module, students will be

- aware of the concerns about safety arising from the use of software-based systems;
- familiar with the major standards for software in safety- related systems;
- capable of applying the concepts of these standards in practical software engineering;
- prepared for advanced study to take responsibility for the development of safety-related systems.

Syllabus

1. OVERVIEW OF SAFETY CONCEPTS — 1 lecture
   Current issues, techniques and standards.
2. HSE GUIDELINES — 3 lectures
   General approach; random and systematic failures. Strategy for safety. Safety "lifecycle".
3. CONTROL AND PROTECTION SYSTEMS — 1 lecture
4. SAFETY REQUIREMENTS — 2 lectures
   Potential dangers. Criteria for safe operation. Reliability and availability requirements. Safety integrity requirements. Analyses of safety and danger. Hazard analysis. Malfunction, unavailability and backup.

5. PROGRAMMING FOR SAFETY AND CORRECTNESS 1 lecture
Software requirements. Influence of programming on safety. Correctness and safety. Quality of software. Confidence and reliability.

6. SYSTEM DESIGN ISSUES 1 lecture
Application of HSE strategies: configuration, quality and reliability.

7. DEVELOPMENT PRINCIPLES 2 lectures
STARTS "life-cycle" model; evolution and change control. Partitioning for safety. Diversity for reliability, Reliability of software tools. Documentation as evidence justifying confidence.

8. STRUCTURAL DESIGN OF SOFTWARE 2 lectures
Program structures. Influence of programming language. Use of subsets of programming languages. Verifiable programs. Design descriptions.

9. LOGICAL AND PHYSICAL DESIGN OF SOFTWARE 2 lectures
Design rules for high-integrity software. Defensive programming. Timing. Input-output handling. Interface details. Interlocking. Device specifics. Resilience.

10. CHECKING PRINCIPLES 2 lectures
Independence, variety and persistence. Quality assurance. Safety integrity analysis. HSE Checklists. Static and dynamic checking.

11. INTELLIGENT INSPECTION 3 lectures
Reviews, walkthroughs, focussed inspections.

12. TESTING 2 lectures
Sampling and exhaustive testing. Test coverage. Unit testing. Integration testing.

13. EVIDENCE FOR CERTIFICATION 1 lecture
Regulatory issues. Legal liability

14. CONCLUSION 1 lecture

Booklist

It is considered essential to purchase the following:

Ian C. Pyle. *Developing Safety Systems*. Prentice Hall International, 1991.

Students are likely to need ready access to the following:

Phil Bennett, editor. *Safety aspects of computer control*. Butterworth-Heinemann, 1993.

HSE. *Programmable Electronic Systems in Safety Related Applications; vol 1 An introductory guide*. Health and Safety Executive, HMSO, 1987. ISBN 0 11 883913 6.

HSE. *Programmable Electronic Systems in Safety Related Applications; vol 2 General Technical Guidelines*. Health and Safety Executive, HMSO, 1987. ISBN 0 11 883906 3.

Felix Redmill and Tom Anderson, editors. *Safety-critical systems*. Chapman and Hall, 1993.

The following should be consulted for different approaches or for further information:

SC65A (Secretariat) 122. *Software for computers in the Application ofindustrial Safety-Related Systems*. International Electro-technical Commission, November 1991.

SC65A (Secretariat) 123. *Functional Safety of ElectricallElectroniclprogrammable electronic systems; generic aspects; Part 1, General Requirements*. International Electro- technical Commission, January 1992.

Ministry of Defence. *Hazard Analysis and Safety Classification of the Computer and Programmable Electronic System Elements of Defence Equipment*. MoD Int Def Stan 00-56 / Issue 1.

Ministry of Defence. *The Procurement of Safety Critical Software in Defence Equipment, Part 1: Requirements*, 1991. MoD Int Def Stan 00-55 (Part 1) / Issue 1.

Ministry of Defence. *The Procurement of Safety Critical Software in Defence Equipment, Part 2: Guidance*,
4.1 1991. MoD Int Def Stan 00-55 (Part 2) / Issue 1.

CS43310 Algebraic and Tabular Documentation Methods for Software

General Description

This module examines algebraic techniques for modelling computing systems. The tabular techniques of D. Parnas are studied.
The lectures for this module will be delivered by staff of The University of Wales, Swansea using the video link.

Commitment	Number of lectures	20
	Number of workshops	10
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
Other staff	Prof. Tucker of UW Swansea	
Pre-requisites	C332(h)/**CS33210**	
Co-requisites	None	
Incompatibilities	Only available to students registered for Masters in Software Engineering	
Assessment	Assessed coursework 10% Written exam - 90%	
Timing	This module is offered only in Semester 2	

Aims

To apply algebraic techniques for modelling computing systems. To apply tabular techniques of D. Parnas for documenting computing systems

Objectives

Syllabus

1. Product life cycle
   Documentation for each stage of the cycle.
2. Requirements in design stages
   Product features specifications; system specifications; and program plans.
3. Testing and root causes of errors
4. Formal methods for analysis of products
   Algebraic modelling methods. Equational specification methods.
5. Formal methods for documentation of products.
   Parnas tables. Transformation of tables. Recursive tables.
6. States versus commands in system documentation
7. Case studies
   Pocket calculators; word processors; general user interfaces.

Booklist

It is considered essential to purchase the following:

A. Wilder and J.V. Tucker. Representing systems using tables - course notes. Technical report, Department of Computer Science, University of Wales Swansea, 1994.

Students are likely to need ready access to the following:

4.1 D. Parnas and J.I. Zucker. 10 research papers on tables.

CS46010 AI Problem Solvers

General Description

This course deals with the implementation of advanced software tools. It builds on C362(h)/**CS36210** giving practical examples in Lisp of common AI techniques and strategies. It will provide an understanding of some important contemporary AI techiques and their applications. At the end of the course, the student should be able to employ such techniques to practical problem-solving tasks.

Commitment	Number of lectures	0
	Number of workshops	14
	Number of practicals	5
Co-ordinator	Dr Fred Long	
Other staff	Not yet known	
Pre-requisites	C362(h)/**CS36210** or equivalent experience	
Co-requisites	None	
Incompatibilities	None	
Assessment	Assessed coursework - 50% Written exam - 50%	
Timing	This module is offered only in Semester 1	

Aims

This course deals with the implementation of advanced software tools. It builds on C362(h)/**CS36210** giving practical examples in Lisp of common AI techniques and strategies. It will provide an understanding of some important contemporary AI techiques and their applications. At the end of the course, the student should be able to employ such techniques to practical problem-solving tasks.

The course will be run as a number of seminars and workshops and will be heavily based on the Norvig book.

Objectives

On successful completion of this module, students should:

- be able to read and understand complex Lisp programs
- have enough experience of Lisp to be able to build their own AI programs in Lisp
- understand the terms and techniques used in AI problem solving
- be able to build software tools using AI techniques

Syllabus

1. OVERVIEW OF COMMON LISP — 2 weeks
   What makes Lisp different. Introduction to the main features of the language and the local Lisp environments.
2. GPS - THE GENERAL PROBLEM SOLVER — 2 weeks
   Description, implementation, example applications, further developments of GPS.
3. PATTERN MATCHING AND SEARCH — 3 weeks
   Segment pattern matching; heuristic tree search; implementing AI search strategies; seaching graphs.

4. CONSTRUCTING ADVANCED SOFTWARE TOOLS — 3 weeks
   Examination of the construction of a variety of software tools including at least some of the following: Interactive interpreter tools; pattern matching tools; rule-based translators; searching tools; theorem provers.

5. EFFICIENCY ISSUES — 2 weeks
   Memoization; compiling one language into another; lazy evaluation; indexing data; low-level efficiency issues.

Booklist

It is considered essential to purchase the following:

Peter Norvig. *Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp*. Morgan-
4.1 Kaufmann, 1992.

CS46310 Model-Based Reasoning for Physical Systems

General Description

A problem at the heart of the research agenda for artificial intelligence is how to reason about the physical world. This module assesses the adequacy of models for this domain and aims to give students an understanding of the issues involved in effectively modelling and reasoning about physical systems. The course will be organised around a number of key real world application domains and will show how the requirements of knowledge and software reuse lead to the need for both compositional models and multiple models of phenomena.

Commitment	Number of lectures	20
	Number of workshops	3
	Number of practicals	3 x 1-hour
Co-ordinator	Dr. Fred Long	
Other staff	Not yet known	
Pre-requisites	**CS46010**	
Co-requisites	None	
Incompatibilities	None	
Assessment	Assessed coursework - 20%	
	Written exam - 80%	
Timing	This module is offered only in Semester 2	

Aims

A problem at the heart of the research agenda for artificial intelligence is how to reason about the physical world. This module assesses the adequacy of models for this domain and aims to give students an understanding of the issues involved in effectively modelling and reasoning about the physical systems. The course will be organised around a number of key real world application domains and these will be used to focus on specific problems that arise in modelling real systems, to show how existing techniques can be used and where such techniques prove inadequate.

Objectives

On successful completion of this module students should:

- be capable of assessing the adequacy of models for effectively reasoning in this domain;
- understand the consequences of different ontological choices;
- be aware of different models of behaviour, and methods for reasoning about change;
- understand how the requirements of knowledge and software reuse lead to the need for compositional modelling and multiple models of phenomena;
- have detailed knowledge of attempts to model behaviour in specific electrical and spatial domains;
- understand the issues involved in automating the modelling process;
- understand the relationship between model-based reasoning and non-monotonic reasoning, and the practical issues that arise out of building model-based systems;
- appreciate how model-based reasoning can be used in diagnosis systems.

Syllabus

1. Introduction — 1 lectures
   The problem with rule-based approaches. Overview of: reasoning from first principles; component libraries; formalisation and automated modeling; qualitative descriptions of behaviour; domain-independent model-based problem solvers.
2. Ontologies — 3 lectures
   Formal languages for conceptual modelling. Component based modelling: component models and interaction modelling. Context-free models. Processes. Bond graphs. Causality. Ontologies for modelling fluids, spatial occupancy and topology.
3. Behaviour Models — 3 lectures
   Difficulties of numeric modelling. Qualitative values, constraints and ambiguity. Order of magnitude reasoning. Interval reasoning.
4. Case Study - Spatial reasoning about mechanisms — 2 lectures
   Mechanics and kinematics. Topological and metric information. Configuration spaces, qualitative reasoning and occupancy arrays.
5. Reasoning About Change — 3 lectures
   Evolving qualitative states. State transitions, ambiguity and multiple time scales. Non-linear systems and phase portraits.
6. Compositional and Multiple Modelling — 2 lectures
   Model properties, assembling model fragments, multiple models and model selection. Logical relations between relational models.
7. Model-Based and Non-Monotonic Modelling — 3 lectures
   Non-monoticity. Persistence. Minimum violation of normality. Default logic. Truth maintenance. Assumption-based truth maintenance.
8. Model-Based systems and Diagnosis — 3 lectures
   Tasks and problems: diagnosis, repair, conceptual and innovative design. failure modes and effects analysis, sensor placement and selection, task generation and testing. Diagnosis: basic concepts, fault models, complexity issues, hierarchies, conflicts and focusing. Managing multiple models.

Booklist

It is considered essential to purchase the following:

4.1 Dept. Collected papers on modelling physical systems. Department of Computer Science, UWA.

CS48110 Professional Studies II

General Description

This module is concerned with marketing, in its broadest sense, in the software industry. It covers the techniques used by the industry to identify and characterise its potential customers and the products and services that they require and the ways in which a software company sets about positioning itself in the market place. It also covers the procurement process, from the point of view both of the procurer and the supplier, and the nature of contracts for the supply of both packaged and bespoke software.

Commitment	Number of lectures	20
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
Other staff	To include lecturers from industry	
Pre-requisites	None	
Co-requisites	None	
Incompatibilities	Only available to students registered for Masters in Software Engineering	
Assessment	Assessed coursework - 100%	
Timing	This module is offered only in Semester 1	

Aims

The aims of this module are to ensure that students understand the central role that commercial issues play in the software industry and to familiarise students with practical aspects of sales, marketing and procurement. The emphasis throughout is on company-to-company relationships rather than on company-to-consumer ones.

Objectives

On successful completion of this module, students should be able to participate in:

- the production of invitations to tender for the supply of software and the evaluation of tenders received;
- the production of proposals in response to an invitation to tender;
- contract negotiations;
- planning of industry sector surveys and marketing initiatives.

They will also understand the legal and economic framework in which public procurement takes place and be familiar with the evolution of defence procurement strategies.

Syllabus

1. MARKETING OF SOFTWARE — 5 lectures
   Estimating the size of a market: sources of data. The marketing mix: retail marketing v. company-to-company marketing; marketing of goods v. marketing of services. The effects of business cycles with respect to sales of capital goods. Market positioning.
2. THE TENDERING PROCESS FOR BESPOKE SOFTWARE — 4 lectures
   Invitations to tender; the tender list; preparation of proposals; evaluation of proposals. Costing and pricing. Single tender situations.

3. PUBLIC PROCUREMENT 6 lectures
   The legal framework; the EC directives. Procurement models used by various public bodies. The problems encountered with large public software procurements. The Downey Report and its successors; the reports of the National Audit Office. The NATO model. Experimental models. The CCTA. Total Procurement Model.
4. CONTRACTS 5 lectures
   Fixed price, firm price, maximum price, cost plus, and target cost incentive fee. Appropriate situations for each type of contract. Substantive contents of contracts for hardware and software procurement and maintenance.

Booklist

There are no books that cover anything more than a small proportion of the material. Copies of relevant reports
4.1 and other special material not easily available will be placed in the library.

CS49060 MEng Project

General Description

This module consists of a major piece of development work carried out by the student. It will include identification of requirements, analysis of the market, cost analysis, design, implementation and evaluation. In appropriate circumstances, two students may be allowed to undertake a project jointly.

Commitment	Number of lectures	6
	Number of workshops	0
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
Other staff	All	
Pre-requisites	None	
Co-requisites	None	
Incompatibilities	Only available to students registered for the degree of MEng in Software Engineering.	
Assessment	Assessed coursework - 100%	
Timing	This module extends over both Semester 1 and Semester 2	

Aims

MEng students are required to do a 60 credit major project as part of their undergraduate experience.

The purpose of the CS49060 project is to give students the opportunity to bring together the practical skills and theoretical understanding that they have acquired, by developing a complete product from the stage of specifying requirements through to demonstrating that the product does indeed satisfy those requirements. This process involves market analysis, producing appropriate documentation, including documentation of the design decisions made, and product evaluation.

Objectives

The objectives of this course are for the student to complete a substantial piece of work, under the direction of a supervisor but demonstrating self discipline, organisation and initiative.

The projects typically involve developing a piece of software from an initial statement of requirements through market analysis, costing, specification and design stages, to successful implementation, testing, and product evaluation. Some projects may include a hardware element.

A list of suggested projects is maintained by the MEng co-ordinator. Projects are normally chosen from this list but we welcome students' own suggestions, especially those arising from industrial year experience.

Syllabus

1. FORMAL LECTURES — 6 lectures
   There are a small number of formal lectures associated with the project.
2. WRITTEN MATERIAL
   Various written material giving guidance on carrying out the project, assessment issues, and the project submission is provided by the department. Students are expected to spend perhaps twenty hours in an average week on the project and to see their supervisor once a week. This course is assessed on the written report submitted by the student. Reports must be submitted by the end of the first week in May. Students will also be required to give a presentation of their project in the form of a seminar, which will account for 10% of the assessment.

3. ASSESSMENT
   A mark of at least 40% in CS49060 is required for an honours degree.

Booklist

4.1 Not appropriate.