Contents

Modules and Units in Computer Science **2**

Degree Schemes **3**

Descriptions of Modules and Units **11**

CS15010 Introduction to Computer Hardware **12**

CS15110 Computer Architecture **14**

CS21020 Program Design, Data Structures and Algorithms **17**

CS22110 The Software Development Lifecycle **20**

CS22210 Operating Systems and Concurrency **22**

CS22510 Case Studies in System Design **24**

CS23110 Small Real-Time Systems **25**

CS23710 C and UNIX Systems Programming **28**

CS25010 Computer Communications **31**

CS25410 Digital System Design **33**

CS27220 Relational Database Systems **35**

CS28010 Personal Communication Skills **37**

C321(h)* Interactive Computer Graphics **39**

C332(h)* Advanced Software Engineering **41**

C333(h)* Formal Methods in Software Engineering **43**

C342(h)* Topics in Programming Languages **45**

C350(h)* Telematics **47**

C362(h)* Artificial Intelligence Concepts **49**

C363(h)* Artificial Intelligence Systems **51**

C364(h)* Industrial Robotics **53**

C374(h)* Information Systems Analysis and Design **55**

C375(h)* Database Management Systems **57**

C380(h)* The Professional Engineer in Society **59**

C390(+)* Major Project **61**

C391(h)* Minor Project **63**

Modules and Units in Computer Science

This session (1994-5) sees the introduction of modularisation into Part II. Students entering their second year do so under modularisation, whereas students entering their third year are still under the unit scheme.

The word 'module' refers to courses offered under modularisation. The words 'unit' and 'half unit' refer to courses offered under the units scheme. These are only available to final year students. Students entering their second year will not be able to take third year units.

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.

There are no third year full units. Courses with the suffix '(h)' are half units, and C390(+) counts as one and a half units.

Students entering their third year this session may take some modules, subject to the rules set out below. In this case, a 20 credit module will correspond to a full unit, and a 10 credit module to a half unit. However, level 1 and level 2 modules will not count as starred units (see under individual degree scheme descriptions).

The Computer Science department will consider *appropriate* units 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 and units, 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 **C390(+)** (the major project) does not reach 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"*.

2.1

Degree Schemes

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

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.

Students must take:

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

CS23110, **CS27220** 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 a total of 6 units in Computer Science. At least 4 of these must be starred units.

Students must take:

- **C332(h)**, **C362(h)**, **C374(h)**, **C390(+)**;
- **CS23110** if they have not previously taken **C231(h);**
- **CS15010** if they have not previously taken **C253(h)** or 1B Hardware;
- **CS27220** if they have not previously taken **C272**.

2.1

BEng Software Engineering

Students entering the second taught year

Students must take 120 credits in Computer Science.
Students must take:

- **C21020**, **C22110**, **CS22210**, **CS22510**, **CS23110**, **CS23710**, **CS25010**, **CS27220**, **CS28010**;
- **CS15010** if they have not already done so;
- 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 6 units with Computer Science.
Students must take:

- **C332(h)**, **C333(h)**, **C342(h)**, **C350(h)**, **C362(h)**, **C374(h)**, **C380(h)**, **C390(+)**;
- sufficient additional units chosen from the following. **C321(h)**, **C363(h)**, **C364(h)**, **C375(h)**.

MEng Software Engineering

Students entering the second taught year

Students must take 120 credits in Computer Science.
Students must take:

- **C21020**, **C22110**, **CS22210**, **CS22510**, **CS23110**, **CS23710**, **CS25010**, **CS27220**, **CS28010**;
- **CS15010** if they have not already done so;
- 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 6 units in Computer Science.
Students must take:

- **CS25410**, **C321(h)**, **C332(h)**, **C333(h)**, **C342(h)**, **C350(h)**, **C362(h)**, **C363(h)**, **C364(h)**, **C374(h)**, **C380(h)**;
- **CS15110** if they did not take 1B Hardware;
- one of the following if they require an additional module. **MA22110**, **MA22210**, **MA23110**, **MA23310**.

2.1

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.
Students must take:

- **CS21020**, **CS22110**, **CS22210**, **CS27220**, **CS28010**;
- **CS15010** if they have not already done so.

Students entering the third taught year

Students must take 4 units in Computer Science and the equivalent of 2 units in their chosen language.
Students must take:

- **C332(h)**, **C362(h)**, **C374(h)**, **C380(h)**, **C390(+)**;
- **C231(h)** if they have not already done so;
- one of the following if they require an additional half unit. **C333(h)**, **C342(h)**, **C363(h)**, **C364(h)**, **C350(h)**, **C375(h)**.

BSc Accounting and Computer Science

Students entering the second taught year

Students must take **CS21020**, **CS22110**, **CS22210** and **CS27220**. A futher 60 credits will be taken in Accounting.

Students entering the third taught year

Students must take 3 units in Computer Science and the approved courses in Accounting.
Students:

- must take **C332(h)** and **C374(h)**;
- are recommended to take **C362(h)**;
- may choose any other Computer Science half units to make their total of 3 units.

2.1

Joint Honours in Computer Science and Another Subject

Students entering the second taught year

Students must take **CS21020**, **CS22110** and **CS22210** plus a further 20 credits of Computer Science at level 2.

Students entering the third taught year

Students must take **C332(h)** plus a further 2.5 (starred) units of Computer Science. 2.1

Information Science

Students entering the second taught year

2.1 In addition to modules required by the Department of Information and Library Studies students must take **CS22110**, **CS25010** and **CS27220**.

Descriptions of Modules and Units

The rest of this book provides details details for each modules and unit offered by the Department.

CS15010 Introduction to Computer Hardware

Commitment	Number of lectures	24
	Number of workshops	0
	Number of practicals	10 x 2 hours
Co-ordinator	Dr. Fred Long	
Pre-requisites	None	
Co-requisites	**CS11010** or **CS10210**	
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.

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

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

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

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

CS15110 Computer Architecture

Commitment	Number of lectures	24
	Number of workshops	-
	Number of practicals	10 x 2 hours
Co-ordinator	Dr. Fred Long	
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.

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

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

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

2.1 M68000 users manual. Motorola.

CS21020 Program Design, Data Structures and Algorithms

Commitment	Number of lectures	48
	Number of workshops	8
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
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 9X.

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

Booklist

It is considered essential to purchase the following:

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

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.

Grady Booch and Doug Bryan. *Software Engineering with Ada*. Benjamin/Cummings, Redwood City, California, 3rd. edition, 1994. (Earlier editions are also available in the library).

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

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++.

CS22110 The Software Development Lifecycle

Commitment	Number of lectures	18
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
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.

CS22210 Operating Systems and Concurrency

Commitment	Number of lectures	24
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
Pre-requisites	None	
Co-requisites	**CS21020**	
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.

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 5 lectures
   Different types of resources. Deadlock prevention and avoidance. Memory management. Segmentation, paging. Discard algorithms, thrashing. Associative stores.
4. PROGRAM AND HUMAN INTERFACES 2 lectures
   Operating system services. Human interface, command languages. Operating systems security.

5. INPUT-OUTPUT 3 lectures
   Input-output services. Filing systems, disc space management.
6. DISTRIBUTED SYSTEMS 2 lectures
   The need for distributed systems, tightly-coupled and loosely coupled systems, the problems which they present to the operating system.
7. SECURITY 3 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.

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.

CS22510 Case Studies in System Design

Commitment	Number of lectures	24
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
Pre-requisites	**CS22210**	
Co-requisites	**CS21020**, **CS22110**	
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

2.1 Not appropriate.

CS23110 Small Real-Time Systems

Commitment	Number of lectures	24
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
Pre-requisites	**CS15010**	
Co-requisites	**CS21020**, **CS22210**	
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 Pyle. *Developing Safety Systems - A Guide Using Ada*. Prentice Hall, 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 Real-Time Systems*. Wiley, 1992.

2.1

CS23710 C and UNIX Systems Programming

Commitment	Number of lectures	24
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
Pre-requisites	**CS22210**	
Co-requisites	**CS21020**	
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.

K. Haviland and H. Salama. *UNIX Systems Programming*. Addison Wesley, 1987.

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.

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.

2.1

CS25010 Computer Communications

Commitment	Number of lectures	24
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
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

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

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. Coax, 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

Students are likely to need ready access to the following:

David M. Piscitello and A. Lyman Chapin. *Open Systems Networking, TCP/IP and OSI*. Addison Wesley, 1993.

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.

2.1

CS25410 Digital System Design

Commitment	Number of lectures	14
	Number of workshops	0
	Number of practicals	10 x 2 hours
Co-ordinator	Dr. Fred Long	
Pre-requisites	**CS15010**	
Co-requisites	**CS21020**	
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 ORCAD.
- A detailed working knowledge of the ORCAD 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; What is ORCAD.
2. BUILDING BLOCKS — 2 lectures
   Sequential logic devices; Counters; Shift-registers; Multiplexers.
3. ORCAD — 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. ORCAD — 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:

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

CS27220 Relational Database Systems

Commitment	Number of lectures	48
	Number of workshops	8
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
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 extends over both Semester 1 and Semester 2	

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;
- discuss the principles underlying human-computer interface design.

Syllabus

1. RELATIONAL MODELLING — 9 lectures
   Entities and relationships. Connection traps. The design of relations. Transformation of E-R model into Relational Schema. Data normalisation. Validating a design.
2. IMPLEMENTING A DATABASE USING ORACLE — 3 lectures
3. UNDERLYING FILE STRUCTURES — 2 lectures
4. 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.

5. INTERFACES AND TOOLS 4 lectures
   The Forms environment SQL in applications programs The Report Writer
6. RELATIONAL DATABASE DESIGN METHODOLOGY 6 lectures
   An overview of design techniques. Overview of the database design process. First level design. Second level design.
7. DATABASE SYSTEM CONCEPTS 3 lectures
8. DATA MODELS 2 lectures
9. THE RELATIONAL DATA MODEL AND ORACLE 9 lectures
   Relational algebra statements in SQL EXISTS, NOT EXISTS and MINUS. If a condition is not fulfilled by all tuples in a relation, then it must be false for at least one tuple. Universal quantification in SQL Relational algebra and its extensions. Set operations. The relational calculus. Relational calculus query languages: QUEL, QBF and Prolog.
10. HUMAN COMPUTER INTERFACES 6 lectures
    Users and usability - the importance of the user interface. Introduction to issues in human-computer interaction. Interface design guidelines.

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.

F. Maddix. *Human-Computer Interaction – Theory and Practice*. Ellis-Horwood, 1990.

A manual on the software system used in the course. To be advised.

F.D. Rolland. *Relational Database Management with ORACLE*. Addison-Wesley, 1989.

S. Stanczyk. *Theory and Practice of Relational Databases*. Pitman, 1990.

2.1

CS28010 Personal Communication Skills

Commitment	Number of lectures	15
	Number of workshops	0
	Number of practicals	5 x 1 hour
Co-ordinator	Dr. Fred Long	
Pre-requisites	None	
Co-requisites	**CS21020**	
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's 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;
- prepare a good curriculum vitae;
- 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.

2.1

C321(h)* Interactive Computer Graphics

Commitment	Number of lectures	24
	Number of workshops	0
	Number of practicals	6 x 1 hour
Co-ordinator	Dr. Fred Long	
Pre-requisites	C210	
Co-requisites	**CS23710** (or C237(h) in previous years) or a basic knowledge of the C language	
Incompatibilities	None	
Assessment	Assessed coursework - 20% Written exam - 80%	
Timing	This half module is offered only in Term 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;
- have constructed and interacted with a virtual reality environment.

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. 6 lectures and 2 practicals
   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.

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.

2.1

C332(h)* Advanced Software Engineering

Commitment	Number of lectures	22
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
Pre-requisites	C210, C220	
Co-requisites	None	
Incompatibilities	None	
Assessment	Assessed coursework - 20%	
	Written exam - 80%	
Timing	This half module is offered only in Term 1	

Aims

This half unit 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).

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, A.N. Earl, and J.A McDermid. *Software Engineering Environments: Automated Support for Software Engineering*. The McGraw-Hill International Series in Software Engineering. McGraw-Hill, 1992.

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.

2.1

C333(h)* Formal Methods in Software Engineering

Commitment	Number of lectures	22
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
Pre-requisites	**C332(h)**	
Co-requisites	None	
Incompatibilities	None	
Assessment	Assessed coursework - 20%	
	Written exam - 80%	
Timing	This half module is offered only in Term 2	

Aims

This half unit aims to develop more rigorously the formal methods introduced in **C332(h)** 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.

2.1

C342(h)* Topics in Programming Languages

Commitment	Number of lectures	22
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Fred Long	
Pre-requisites	C210	
Co-requisites	None	
Incompatibilities	None	
Assessment	Assessed coursework - 20%	
	Written exam - 80%	
Timing	This half module is offered only in Term 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.

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

C350(h)* Telematics

Commitment	Number of lectures	24
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
Pre-requisites	**CS25010** or C250(h)	
Co-requisites	**CS**23710 (or C237(h) in previous years)	
Incompatibilities	None	
Assessment	Assessed coursework - 20%	
	Written exam - 80%	
Timing	This half module is offered only in Term 2	

Aims

This half unit 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 half unit, 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/Fax/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.

2.1

C362(h)* Artificial Intelligence Concepts

Commitment	Number of lectures	22
	Number of workshops	0
	Number of practicals	6 x 1-hour
Co-ordinator	Dr. Fred Long	
Pre-requisites	C210, C220	
Co-requisites	None	
Incompatibilities	None	
Assessment	Assessed coursework - 20%	
	Written exam - 80%	
Timing	This half module is offered only in Term 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, 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:

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

C363(h)* Artificial Intelligence Systems

Commitment	Number of lectures	12
	Number of workshops	6
	Number of practicals	6 x 1-hour
Co-ordinator	Dr. Fred Long	
Pre-requisites	**C362(h)**	
Co-requisites	None	
Incompatibilities	None	
Assessment	Assessed coursework - 20%	
	Written exam - 80%	
Timing	This half module is offered only in Term 2	

Aims

This module addresses the issues concerning the development of knowledge-based systems. This necessarily involves consideration of:

- the characteristics of the problem domain;
- knowledge aquisition methods;
- evaluation of a suitable knowledge engineering toolkit;
- software engineering methodology for knowledge-based systems;

It is assumed that students undertaking this module will spend considerable time outside of lectures following of course of directed reading, consisting of technical reports, technology surveys and case studies. The module seminars will be a forum for discussing issues arising from this wider reading, and their relatationship to issues discussed in the lectures.

Objectives

On successful completion of this module students should:

- be aware of the implications of a problem domain for the selection of knowledge aquisition methodology, knowledge engineering toolkit, software development methodology, and system evaluation strategy;
- understand the differences between the main classes of knowledge engineering toolkit, including cost-benefit considerations, representation and reasoning facilitites, and performance issues;
- understand the different techniques and methodologies available for knowledge aquisition, and their relative strengths and weaknesses;
- appreciate the problem of software engineering for knowledge-based systems;
- have had practical exposure to a range of knowledge engineering toolkits: including an expert system shell; a loosely coupled toolkit; and a high-end intergrated toolkit;

Syllabus

1. OVERVIEW OF AI SYSTEMS — 1 lecture
   Reasons for building knowledge-based systems. Overview of issues in knowledge-based system development. Survey of typical knowledge-based systems.
2. CURRENT EXPERT SYSTEM TECHNOLOGY — 2 lectures and 3 practicals
   Overview of rule-based expert systems. Introduction to a commercial expert system shell. Representation, inference, control flow, and interface issues. A first attempt at constructing an expert system. Rule-based expert system shell programming.
3. CLASSIFYING THE PROBLEM DOMAIN — 1 lecture
   Classification problems: monitoring, prediction, diagnosis and decision classification. Construction problems: design, planning, simulation and repair. Composite problems: process control, troubleshooting and instruction. Case studies. seminar.
4. KNOWLEDGE ENGINEERING TOOLKITS — 4 lectures and 3 practicals
   Classifying of knowledge engineering toolkits. Artificial Intelligence languages, loosely coupled toolkits, integrated toolkits, high-end integrated toolkits, expert system shells. Case studies. Implications of the problem domain classification for toolkit selection.
5. KNOWLEDGE ACQUISITION — 2 lectures
   Knowledge acquisition and knowledge elicitation. Problems with knowledge elicitation. Stages in knowledge acquisition. Goal decomposition techniques. Classification techniques. Automating the knowledge acquistion process.
6. THE KNOWLEDGE ENGINEERING LIFECYCLE — 2 lectures
   Waterfall and spiral models. Feasibility study; requirements specification; system design; system construction; testing. Knowledge-based system for software engineering. One seminar on the role of software engineering in AI.

Booklist

Students are likely to need ready access to the following:

M. Bramer. *Practical Experience in Building Expert Systems*. Wiley, 1990.

P. Harmon and B. Sawyer. *Creating Expert Systems for Business and Industry*. Wiley, 1990.

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

Pedersen. *Expert Systems Programming*. Wiley, 1989.

E. C. Payne and R. Macarthur. *Developing Expert Systems: a knowledge engineer's handbook for rules and objects*. Wiley USA, 1990.

C. Price. *Knowledge Engineering Toolkits*. Ellis Horwood, 1989.

2.1

C364(h)* Industrial Robotics

Commitment	Number of lectures	22
	Number of workshops	0
	Number of practicals	3 x 2 hours
Co-ordinator	Dr. Fred Long	
Pre-requisites	C210	
Co-requisites	**C362(h)**	
Incompatibilities	None	
Assessment	Assessed coursework - 20%	
	Written exam - 80%	
Timing	This half module is offered only in Term 1	

Aims

This half unit 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 half unit, 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 — 2 lectures and 1 practical
   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 and 2 practicals
   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 1 lecture
   Robot modelling; solid body systems; simulation.
5. SENSING 4 lectures
   Sensory modalities; smart sensors; software implications; the perception problem.
6. COMPLIANCE 1 lecture
   Active and passive compliance. The software implications of active compliance; simple control algorithms; the problem of specification.
7. IMPLICIT ROBOT PROGRAMMING 4 lectures
   World modelling; task specification; path planning; grasp planning; sequencing; program synthesis. The AL, RAPT and AUTOPASS systems.
8. THE ERROR PROBLEM 2 lectures
   The practical requirements for error handling. The monitor-detect-plan-recover model. Continuous error and the control approach.
9. BEHAVIOUR BASED ROBOT PROGRAMMING 2 lectures
   Outline of the concept; consideration of its benefits; examination of its contribution.
10. CASE STUDY 1 lecture
    Study of the **InFACT** machine as an example of a market driven flexible automatic assemble machine.

Booklist

Students are likely to need ready access to the following:

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

C374(h)* Information Systems Analysis and Design

Commitment	Number of lectures	20
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
Pre-requisites	None	
Co-requisites	**CS27210** (or C272(h) as a pre-requisite)	
Incompatibilities	None	
Assessment	Assessed coursework - 20%	
	Written exam - 80%	
Timing	This half module is offered only in Term 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 1 lecture
   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 transactions.
9. OBJECT ORIENTED DATA MODELLING 4 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.

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.

2.1

C375(h)* Database Management Systems

Commitment	Number of lectures	20
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
Pre-requisites	None	
Co-requisites	**CS27220** (or C272 as a pre-requisite)	
Incompatibilities	None	
Assessment	Assessed coursework - 20%	
	Written exam - 80%	
Timing	This half module is offered only in Term 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.

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

C380(h)* The Professional Engineer in Society

Commitment	Number of lectures	20
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Dr. Fred Long	
Pre-requisites	None	
Co-requisites	**CS33210** or by arrangement with the department	
Incompatibilities	None	
Assessment	Assessed coursework - 100%	
Timing	This module extends over both Term 1 and Term 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
   The law relating to copyright, patents and trademarks.

4. HEALTH, SAFETY AND ENVIRONMENTAL 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.

2.1

C390(+)* Major Project

Commitment	Number of lectures	4
	Number of workshops	0
	Number of practicals	0
Co-ordinator	Dr. Horst Holstein	
Pre-requisites	This course is for single honours Computer Science students, and for BEng students	
Co-requisites	None	
Incompatibilities	**C391(h)**	
Assessment	Assessed coursework - 100%	
Timing	This module extends over both Term 1 and Term 2	

Aims

Single Honours Computer Science students are required to do a major project, which counts for one and a half units, as part of their undergraduate experience.

(Joint Honours students are not normally encouraged to do a C390(+) project, as opposed to a **C391(h)** project because this would leave them short of taught units.)

The purpose of the C390 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 unit co-ordinator. Projects are normally chosen from this list but we welcome students' own suggestions, especially those arising from industrial year experience.

Syllabus

1. There are a small number of formal lectures associated with the project.
2. 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.

A mark of at least 40% in C390(+) is required for an honours degree.

Booklist

2.1 Not appropriate.

C391(h)* Minor Project

Commitment	Number of lectures	4
	Number of workshops	0
	Number of practicals	0
Co-ordinator	Dr. Horst Holstein	
Pre-requisites	This course is for joint honours Computer Science students	
Co-requisites	None	
Incompatibilities	C390(+)	
Assessment	Assessed coursework - 100%	
Timing	This module extends over both Term 1 and Term 2	

Aims

This unit is intended for joint honours students who wish to undertake project work as part of their final year course. All details are like C390(+), 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 a full unit 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 a half unit 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 C390 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. There are a small number of formal lectures associated with the project.
2. 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

2.1 Not appropriate.