Contents

Postgraduate Diploma/M.Sc. Conversion Course **2**

Course Modules **3**

CSM0110 Foundation Studies **4**

CSM0220 Software Development, Data Structures and Algorithms I **6**

CSM0310 Software Development, Data Structures and Algorithms II **8**

CSM0510 Computer Architecture **11**

CSM2110 The Software Development Lifecycle **14**

CSM2210 Operating Systems and Concurrency **16**

CSM2510 Case Studies in System Design **18**

CSM7210 Introduction to Database Systems Design **19**

CSM8010 The Professional Engineer in Society **21**

Additional Elements of the MSc/Diploma Course **23**

Postgraduate Diploma/M.Sc. Conversion Course

Introduction

The University Diploma/M.Sc. postgraduate course in Computer Science is intended for people who have already graduated in other disciplines. It leads to a University Diploma or an M.Sc. degree. It consists of an eight month taught course, after which students undertake further practical work until the end of June, when a Diploma may be awarded. Alternatively, students who have achieved an appropriate level in the taught course can then, if they wish, proceed to a substantial project which is submitted as a dissertation for the M.Sc. degree.

The course organisation is described below. This is followed by a detailed syllabus for the eight month taught course.

The Science and Engineering Research Council has accepted the M.Sc. course as suitable for tenure of its Advance Course Studentships. Some students on the Diploma/M.Sc. course may be eligible for financial assistance from other sources.

Course Organisation

The course is organised as follows:

- **September** Foundation Module
- **October through April** Further Course Modules
- **Early May** Unix and C
- *either* **May through June** Advanced Practical Assignments
- *or* **May through August** M.Sc. Project and Dissertation

Commitment

In addition to lectures, tutorials, seminars and supervised practicals, students will normally need to spend time in the library and in workstation rooms. This may involve evening work.

Method of Assessment

Satisfactory completion of course modules and advanced practical assignments will qualify a candidate for a University Diploma in Computer Science. A student who achieves distinction level in the course modules *and* in the advanced practical assignments will qualify for a Diploma with Distinction.

To qualify for an M.Sc. degree in Computer Science, candidates must achieve satisfactory grades in course modules and in an individual project dissertation. For a student to be awarded an M.Sc. with Distinction they must achieve **Distinction Level** in *both* the course modules *and* in the dissertation.

1.3 The course coordinator is Dr. Edel Sherratt.

Course Modules

CSM0110 Foundation Studies

Commitment	Number of lectures	42
	Number of workshops	16
	Number of practicals	15 x 3 hours, 7 x 2 hours
Co-ordinator	Dr. Edel Sherratt	
Pre-requisites	None	
Co-requisites	None	
Incompatibilities	This module is not available to undergraduate students	
Assessment	Assessed coursework 50% Written exam 50%	
Timing	This module is offered in September only	

Aims

This module provides an introduction to computer science by covering a wide variety of topics fundamental to concepts covered in the later modules. Concentrating on software development, the module introduces the idea of problem solving and algorithm design. The programming language Ada is used to implement these designs.

It is not assumed that participants on this module have any prior knowledge in computing. This is an intensive module designed to cover a wide range of material in a short amount of time. In addition to the lectures, a substantial amount of time is spent in tutorials and supervised practicals.

Objectives

On successful completion of the module, students should:

- be familiar with a broad range of introductory topics including computer systems appreciation, mathematics for computing, document preparation and professional issues;
- understand the most basic concepts of software development;
- be familiar with the range of computer equipment available at Aberystwyth, its availability and capability.

Syllabus

1. INTRODUCTORY LECTURES AND PRE-VIEW: MANAGEMENT ISSUES AND PROFESSIONAL CONDUCT 4 lectures
   Introduction to the department and the course. What is computing all about: growing role of computing, software, the need for software engineering, professionalism. Software engineering: requirements, design, implementation, testing, maintenance, quality, need for documentation.
2. COMPUTER SYSTEM APPRECIATION 3 lectures
   Introduces the student to basic computer organisation. Emphasis is placed on the relationships

between hardware, architecture and software. These presentations are illustrated by reference to the computer systems at Aberystwyth.

3. DEPARTMENTAL COMPUTING FACILITIES 3 lectures
This acquaints the student with the computing facilities run by the department. The work is primarily concerned with high performance workstations. The topics covered will include file preparation and manipulation followed by the use of our Ada language system. Use of the electronic mail system will also be covered. The material will be supported by timetabled practical work with assistance from demonstrators.

4. INTRODUCTION TO SOFTWARE DEVELOPMENT 22 lectures
This introduces the idea of problem solving and algorithm design together with subsequent implementation using Ada. The following topics are covered (Number of lectures in brackets): Software crisis, abstraction, algorithms and programs. (1) The design of algorithms and introduction to Ada. (2) The representation of algorithms. (2) Programming constructs, expressions, simple types. (3) Testing - techniques and aids for error detection. (2) Information hiding - introduction to procedures. (4) Abstraction support using packages. (5)

5. MATHEMATICS FOR COMPUTING 5 lectures
This covers various mathematical topics which are important in computing. The following topics are covered (Number of lectures in brackets): Propositional calculus. (1) Boolean algebra and its applications to switching theory. (1) The algebra of sets. (1) Numbers and number systems. (1) Rates of growth of functions. (1)

6. INTRODUCTION TO TELEMATICS: 3 lectures
Telematic applications - mailers, teleworking, the internet.

7. DOCUMENT PREPARATION 2 lectures
This introduces students to the document preparation facilities provided by the main College Computing service. It also introduces Windows which is frequently encountered on small business computers.

Booklist

It is considered essential to purchase the following:

J G P Barnes. *Programming in Ada*. Addison-Wesley, Wokingham, England, 4th. edition, 1993.

J.E.Cooling, N.Cooling, and J.Cooling. *Introduction to Ada*. Chapman and Hall, 1993.

Shelley and Hunt. *Computer Studies - A First Course*. Pitman, 1985.

N.B. Students need only obtain *either* Barnes *or* Cooling, Cooling and Cooling. The latter is better suited for those who have no programing experience.

Students are likely to need ready access to the following:

T. Andreasson and J. Skansolm. *Getting Started with UNIX and X*. Addison Wesley, 1992.

D. Gilly. *Unix in a Nutshel*. O'Reilly and Associates, 1992.

CSM0220 Software Development, Data Structures and Algorithms I

Commitment	Number of lectures	36
	Number of workshops	6 (latter part of semester)
	Number of practicals	6 (first part of semester)
Co-ordinator	Dr. Edel Sherratt	
Pre-requisites	**CSM0110**	
Co-requisites	None	
Incompatibilities	This module is not available to undergraduate students	
Assessment	Assessed oursework 50% Written exam 50% This course is assessed at the end of Semester 1.	
Timing	This module begins during September and continues through Semester 1.	

Aims

This module concentrates on teaching a professional approach to software development. Emphasis is placed on distinguishing between the design of a software product and its implementation. It develops the programming skills of participants by:

- introducing more advanced facilities that are available to the software engineer to improve the robustness, reusability and maintainability of software;
- applying these skills in practice using the Ada programming language;
- introducing the main Abstract Data Types common in software development.

Objectives

The course concentrates on a professional approach to software development. On successful completion of the module, students should:

- have an understanding of the concepts of problem abstraction and program design;
- be capable of realising their design in the Ada programming language producing software that is robust, reusable and maintainable;
- have an understanding of the Unix operating system at a level to support their programming requirements.

Syllabus

1. FURTHER DESIGN AND IMPLEMENTATION OF COMPUTER PROGRAMS 16 lectures
   This module looks at more advanced programming facilities available to the software developer (numbers of lectures in brackets). Composite types - arrays and records. (3) Developing robust programs - illustrated by exceptions. (4) Building abstract data types - case studies. (4) Handling persistent data - advanced input/output and files. (4)

2. UNIX: AN ENVIRONMENT TO SUPPORT COMPONENT REUSE? 2 lectures
   This modules looks at some of the basic UNIX components e.g. *sort*, *uniq* and the pipe.
3. INTRODUCTION TO TELEMATICS: 2 lectures
   Telematic applications - mailers, teleworking, the internet.
4. INTRODUCTION TO TELECOMMUNICATIONS: 3 lectures
   Network protocols, communication technologies, and standards.
5. INTRODUCTION TO THE ANALYSIS OF ALGORITHMS 3 lectures
   O() notation, growth rates. This material will be used as the basis of more detailed discussions of algorithms later in this course and in **CSM0310**.
6. DYNAMIC STORAGE ALLOCATION 4 lectures
   Introduction dynamic allocation (using the heap). Comparisons of bounded and unbounded forms of data structures. Discussion of the failure modes associated with pointers and dynamic allocation.
7. STORING AND RETRIEVING DATA BY KEY 6 lectures
   This problem motivates the discussion of some important data structures and algorithms, the features of various solutions being related to the dimensions of the problem such as the volume of the data to be handled, volatility, and the operations required. Binary searching, trees, a brief look at hashing.
8. ANOTHER FURTHER LOOK AT ABSTRACT DATA TYPES 3 lectures
   Building more general ADTs. Parameterisation by type using the Ada *generic* facility.
9. PRACTICALS 6 practicals
   This module has a heavy practical content which is used to give students actual experience of the techniques that are described in class.

Booklist

It is considered essential to purchase the following:

J G P Barnes. *Programming in Ada*. Addison-Wesley, Wokingham, England, 4th. edition, 1993.

J.E.Cooling, N.Cooling, and J.Cooling. *Introduction to Ada*. Chapman and Hall, 1993.

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

N.B. Students do not need to obtain both these Ada texts. Choose the one whose style you like.

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

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

CSM0310 Software Development, Data Structures and Algorithms II

Commitment	Number of lectures	24 (6 * 4)
	Number of workshops	6
	Number of practicals	0
Co-ordinator	Dr. Edel Sherratt	
Pre-requisites	**CSM0220**	
Co-requisites	None	
Incompatibilities	This module is not available to undergraduate students	
Assessment	Assessed coursework 50%	
	Written exam 50%	
Timing	This module is offered in the early part of Semester 2	

Aims

This module builds on the foundation provided by **CSM0210** and 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 express more naturally their design objectives in code.

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

Objectives

On completion of this module, students should:

- Understand the principles of abstractions and encapsulation as they apply to the design of abstract data types and programs;
- Be able to design and implement significant programs in Ada;
- Understand the importance of the running time behaviour of algorithms and how this is expressed and determined;
- Understand some of important 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;
- Have an appreciation of the importance of program correctness and some of the techniques used to ensure it.

Syllabus

1. STORING AND RETRIEVING DATA IN EXTERNAL STORAGE 5 lectures
   Some techniques used for storing large volumes of data on external media. Performance characteristics of disks versus memory. Using Ada Direct IO to implement persistent random-access storage. Hashing and B-tree organisations.
2. REPRESENTING TEXT 4 lectures
   A simple text editor will be used as the basis of a design case-study showing how the deign decisions concerning the different ways of representing text affect the performance and ease of implementation. String matching algorithms and their performance are examined in the context of the searching operations to be provided by the editor.
3. SORTING 4 lectures
   A look at some important sorting algorithms. Divide and conquer algorithms.
4. REPRESENTING COMPLEX RELATIONSHIPS: GRAPHS AND GRAPH ALGORITHMS 5 lectures
   Terminology. Ways of representing graphs. A look at some graph- related problems and the algorithms used to solve them. Greedy algorithms.
5. THE LIMITATIONS OF ADTS 3 lectures
   What happens when an ADT does almost what is required. An introduction to the concepts of object-oriented programming and design and the object-oriented features of Ada 9x.
6. DESIGNING SOFTWARE IN OTHER LANGUAGES 3 lectures
   The importance of applying good design practices when using other languages. 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.

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

Grady Booch and Doug Bryan. *Software Engineering with Ada*. Benjamin/Cummings, Redwood City, California, 3rd. edition, 1994. (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 L Kruse. *Data Structures and Program Design*. Prentice-Hall, Englewood Cliffs, New Jersey, second edition, 1987. (The first edition is also available from the library).

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

1.6

CSM0510 Computer Architecture

Commitment	Number of lectures	20
	Number of workshops	0
	Number of practicals	10
Co-ordinator	Dr. Edel Sherratt	
Pre-requisites	**CSM0110**	
Co-requisites	None	
Incompatibilities	This module is not available to undergraduate students	
Assessment	Assessed coursework 50%	
	Written exam 50%	
Timing	This module extends over both Semester 1 and Semester 2	

Aims

It is the aim of this module to give a broad overview of computer architecture, and how it influences the execution and performance of software, together with an awareness of low-level programming languages. In addition, students will gain an appreciation of interfacing microprocessor based system to the digital and analogue world.

- Introduction to logic and its hardware implementations.
- Familiarisation with the principal components of a microprocessor based system and their differing roles.
- An appreciation of the interactions of typical components found in a small embedded system.
- An understanding of the principles and techniques involved in the interfacing of such systems to the real world.
- To provide an insight into the internal architecture of microprocessors in general.
- To raise an awareness of the role and intricacies of low-level languages and their relevant application.

Objectives

Students successfully compleating this module will have a broad functional understanding of computer architecture and an awareness of the hardware software interface. In detail, they will be expected to have:

- A sound knowledge of the function and operation of combinational and sequential logic; Gates; Latches; Storage registers; Shift registers; Counters;
- An understanding of the principles and applications of I.C. output types; Totem Poles; Tri-states; Open Collectors;
- A knowledge of the fundamental components found in a typical microprocessor based system and their dependencies; Microprocessors; Memory; I/O devices; Buses; Decoding;

- The ability to produce an outline design for a small embedded microprocessor system together with a detailed memory map, from a given specification;
- A working knowledge of the internal architecture of a typical microprocessor and their roles in the execution of software; MBR; MAR; Control Unit; PSR; IR; ALU;
- An appreciation of low-level language programming; Motorola 6800 assembly language; Assignment instructions, Arithmetic instructions, Logic instructions; Control instructions; Addressing Modes; Role of the stack in subroutines and interrupts;
- An knowledge of the functional operation of Cache memory; Types; Performance considerations;
- An understanding of the role and operation of Direct Memory Access; Control of DMA; Bus Arbitration; Performance Considerations;
- An appreciation of trends past and present in the hardware field; Processor families; Single-Chip processors; Parallel capabilities; Cost/Performance trends.

Syllabus

1. INTRODUCTION — 1 lecture
   Course Organisation; Why study architecture? What is hardware?
2. NUMBER SYSTEMS — 4 lectures
   Logic functions; Notation; Truth Tables; Boolean relationships; De Morgan's Theorem; Implementation; Combinational logic; Sequential logic: RS-Latch; D-Type Latch; JK Flip-Flops; Storage registers; Shift registers; counters.
3. BASIC COMPUTER HARDWARE — 4 lectures
   Principal components of a microprocessor based system; CPU; RAM; ROM; I/O; Buses; Memory organisation; Memory maps; Memory decoding; Bus timing; Synchronous and Asynchronous operation
4. BASIC COMPUTER ARCHITECTURE — 8 lectures
   Bus transactions; Assembly language (6802 based); Compiling; Assembling; Relocatable object code; Subroutines and Stacks; Programme execution; Internal architecture of a microprocessor: MBR, MAR, ALU, IR, PSR, Control Unit; Interrupts; Direct Memory Access; Mass storage devices; Cache memory: Fully associative, Direct mapped.
5. MICROPROCESSOR EVOLUTION — 3 lectures
   Concept of a Microprocessor Family; CISC and RISC; Single-Chip processors; Parallelism; The Transputer; Branch prediction; Pipelining; Trends past and future.
6. INTRODUCTION TO LABORATORY WORK — 2 practicals
   Use of laboratory; Use of test equipment; Use of measuring equipment. Use of bread-boards; Use of power supplies; Use of good wiring practice.
7. COMBINATIONAL LOGIC — 1 practical
   Investigation of logic gates; Synthesis of combinational logic circuits.
8. SEQUENTIAL LOGIC — 1 practical
   Investigation of sequential logic devices; Practical applications.

9. INVESTIGATION OF A SMALL EMBEDDED SYSTEM 6 practicals
   Advanced use of Oscilloscope; Control bus operation; Address bus operation; Data bus operation; Timing considerations.

Booklist

It is considered essential to purchase the following:

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

Students are likely to need ready access to the following:

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

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

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

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

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

CSM2110 The Software Development Lifecycle

Commitment	Number of lectures	18
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Dr. Edel Sherratt	
Pre-requisites	**CSM0110**	
Co-requisites	None	
Incompatibilities	This module is not available to undergraduate students	
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 **CSM7210** contains 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.

CSM2210 Operating Systems and Concurrency

Commitment	Number of lectures	24
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Dr. Edel Sherratt	
Pre-requisites	**CSM0110**	
Co-requisites	**CSM0220**	
Incompatibilities	This module is not available to undergraduate students	
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.

CSM2510 Case Studies in System Design

Commitment	Number of lectures	24
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Dr. Edel Sherratt	
Pre-requisites	**CSM2210**	
Co-requisites	**CSM2110**	
Incompatibilities	This module is not available to undergraduate students	
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

1.4 Not appropriate.

CSM7210 Introduction to Database Systems Design

Commitment	Number of lectures	20
	Number of workshops	0
	Number of practicals	4 x 1 hour
Co-ordinator	Dr. Edel Sherratt	
Pre-requisites	**CSM0110**	
Co-requisites	None	
Incompatibilities	This module is not available to undergraduate students	
Assessment	Assessed coursework - 50%	
	Written exam - 50%	
Timing	This module is offered only in Semester 1	

Aims

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

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.

Syllabus

1. RELATIONAL MODELLING — 8 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 — 3 lectures
   The Forms environment. SQL in applications programs. The Report Writer.

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.

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.

CSM8010 The Professional Engineer in Society

Commitment	Number of lectures	20
	Number of workshops	4
	Number of practicals	0
Co-ordinator	Dr. Edel Sherratt	
Pre-requisites	None	
Co-requisites	**CSM2110**	
Incompatibilities	This module is not available to undergraduate students	
Assessment	Assessed coursework - 100%	
Timing	This module extends over both Semester 1 and Semester 2	

Aims

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

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

Objectives

On successful completion of this module, students should understand:

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

Syllabus

1. ORGANISATIONS AND THEIR STRUCTURES 3 lectures
   Limited companies, private and public; partnerships; sole traders. Special features of limited

companies; responsibilities of directors. Functional organisation and product line organisation. Centralised and decentralised organisations. Management styles. Management by objectives.

2. COMPANY FINANCE 5 lectures
The need for capital; investment and working capital; sources of funds; equity capital and loan capital. Cash flow and its importance. Costing: fixed costs and variable costs; overheads; opportunity costs; depreciation. Problems of cost allocation. Budgeting. Financial accounts: balance sheets, profit and loss accounts, sources and application of funds. Assessment of capital investment. Discounted cash flow analysis.

3. INTELLECTUAL PROPERTY RIGHTS 4 lectures
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:

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

Additional Elements of the MSc/Diploma Course

Unix and C

This part of the course is equivalent to the undergraduate module **CS23710**.

Advanced Practicals or Individual Project

Depending on the results of the examinations and on personal choice, a candidate may proceed either to advanced practical assignments, leading to a University Diploma, or to individual project which is submitted as a dissertation for the M.Sc. degree. A candidate who achieves about 40% in the course modules may proceed towards a University Diploma, and a candidate who achieves about 50% may choose to proceed either towards a Diploma or towards a project for an M.Sc. in Computer Science.

Advanced Practical Assignments

This part of the course consists of two assignments to be completed between May and mid-June. One assignment will provide experience with a database system, and the other will be a substantial Ada assignment.

Individual Project

This part of the course consists of a substantial project, which is submitted as a dissertation to be considered for an M.Sc. degree. Submission will normally be required by the end of August.

Projects will usually be linked to the research activities of the department, which are described in the general departmental brochure. 1.1