Prifysgol Cymru

Aberystwyth

The University of Wales

Department of Computer Science

in association with

Informatics Group Ltd

Postgraduate Diploma/MSc October 2001

Contents

Introduction

This booklet is intended for students on the course for the MSc in Computer Science, offered in Singapore, by the University of Wales, Aberystwyth, in association with Informatics Group Ltd. It describes the aims and objectives of the course, its structure, how it is assessed, and the regulations that govern it. It also contains a detailed description of the individual modules.

This booklet is to be taken in the context of the University of Wales' regulations entitled *Regulations for Master's Degrees by Examination and Dissertation (Modular Structure)*, copies of which are available on request.

The course described in this booklet is the one that we expect to offer to students entering the programme in October 2001 and progressing through it normally, that is, completing the taught part of the programme within 18 months of starting. We reserve the right to change this programme for reasons of *force majeur*; we may also, on occasion, change it in response to requests from the student body as a whole.

If you start the programme in October 2001 but take longer than 18 months to complete the taught part of the course, we cannot guarantee that the programme will be the same, although it is unlikely to change substantially.

Aims and Objectives

The aim of the *MSc in Computer Science* is to provide a Master's level qualification in Computer Science for students who have a Bachelor's degree in a non-computing discipline, some professional experience in industry or commerce, and some experience of using IT. On completion of the course, students will:

- be able to contribute directly to the development of computer systems, in application areas with which they are familiar from their previous qualifications and experience;
- be able to manage the development and operation of information systems of moderate complexity;
- understand the meaning and importance of quality in the context of computer software;
- understand the role of information systems in a business and the importance of non- technical factors in assuring their success.

Course Content and Structure

In order to qualify for the award of an MSc in Computer Science, a student must study modules amounting in total to 180 credits.

The programme consists of the modules listed in Table 1. (The last two digits of the module identifier specify the number of credits associated with the module.)

In addition, students who have little or no previous experience of programming in Java must take, and pass, the module **CO11020:** Introduction to programming. There is no extra fee for taking this module and the credits arising from it do not constitute part of the assessment for the MSc.

The taught modules (i.e. all modules except **COM9060**) are taught over a period of about 18 months. During this period, modules will be delivered intensively over a one or two week period, by staff from Aberystwyth visiting Singapore. The following is the provisional teaching pattern for 10-credit modules:

CO21120:	Data structures, algorithms and software design
CO23010:	Computer architecture
COM1220:	The software development life cycle
COM3110:	The UNIX environment and the C language
COM5010:	Introduction to communications and telematics
COM5610:	The Internet: telematic application design and construction
COM6010:	Artificial intelligence
COM7020:	Databases and data analysis
COM8110:	Software management and the software industry
COM9060:	MSc project (dissertation)

Table 1: The required modules in the scheme

First Saturday:	classes 1400 to 1800
First Sunday:	classes from 0900 to 1230 and 1330 to 1800
Monday:	classes from 1900 to 2200
Wednesday:	the same
Friday:	the same
Second Saturday:	classes 1400 to 1800
Second Sunday:	classes from 0900 to 1230 and 1330 to 1800
Third Friday	possible video-conferencing session from 1930 to 2130
Fifth or sixth Sunday:	written examination (where required), course work due in.

For 20-credit modules the provisional pattern is:

First Saturday:	classes from 1400 to 1800
First Sunday:	classes from 0900 to 1230 and 1330 to 1800
Monday:	classes from 1900 to 2200
Wednesday:	the same
Friday:	the same
Second Saturday:	classes from 1400 to 1800
Second Sunday:	classes from 0900 to 1230 and 1330 to 1800
Monday:	classes from 1900 to 2200
Wednesday:	the same
Friday:	the same
Third Saturday:	classes from 1400 to 1800
Third Sunday:	classes from 0900 to 1230 and 1330 to 1800
Fourth Friday:	possible video-conferencing session from 1930 to 2130
Sixth or seventh Sunday:	written examination (where required), course work due in.

Subject to satisfactory performance in the taught part of the course, students will spend the next nine months or so working on their project and dissertation.

Students are normally expected to take modules in sequence along with the rest of their intake. In this way, the taught part of the course can normally be completed in around 18 months. However, it is realised that work and other commitments, as well as sickness, may sometimes make this difficult or even impossible. A student who needs to defer taking a module should contact the course co-ordinator (Dr Mora McCallum, mom@aber.ac.uk) explaining the reasons for seeking the deferment. This should be done as far as possible in advance of the start of the module in question. If the deferment is approved, the student will be allowed either to attend a later presentation of the module or to attend an alternative module, at the discretion of the University and subject to the overall time constraints described under “Submission dates” below.

Except in cases of *force majeur* (e.g. serious illness), students must attend for examination on the scheduled day unless they have previously made arrangements to take the examination on a different occasion.

Any such arrangements must be agreed at least one week before the date of the examination and will incur an additional fee (see under 'Supplementary Charges' below). Students who fail to present themselves for examination at the agreed time will be treated as having failed; they will be required to resit the examination and will not be eligible for any subsequent resits beyond that one. (See under 'Resits' below.)

Assessment

To qualify for progression to the dissertation stage of the MSc a candidate must obtain:

1. an average of at least 50% over the taught modules;
2. marks of 50% or above in at least 80 credits of the taught modules;
3. no marks below 40%, except for a maximum of 10 credits between 35-39%.

In order to be awarded the MSc a candidate must then achieve a mark of at least 50% in the dissertation.

In order to be awarded the MSc with distinction, a candidate must obtain:

1. an average of not less than 65% over the taught modules;
2. marks of 50% or above in at least 80 credits of the taught modules;
3. no module marks below 40%, except for a maximum of 10 credits between 35-39%;
4. a mark of not less than 70% for the dissertation on its first submission;
5. a weighted average of not less than 70% over the taught modules and dissertation.

Candidates who, at the end of the taught part of the course, have failed to meet the requirements to progress to the dissertation stage will be awarded a University postgraduate diploma provided they have obtained:

1. an average of at least 40% overall over 120 taught credits;
2. marks of 40% or above in at least 80 credits;
3. no marks below 30%.

Note that the marks obtained in **CO11020:** Introduction to programming, by those students who need to take it, do not enter into any of the criteria described above.

A diploma will also be awarded to candidates who achieve the marks necessary for the award of an MSc in the taught modules but who

1. do not wish to proceed to the dissertation phase; or
2. fail to submit a dissertation within the approved time limits; or
3. submit a dissertation that is judged not to be of sufficient quality to merit the award of the MSc and fail to submit a revised dissertation of suitable standard within the approved time limit.

The way in which individual modules are assessed is described under the detailed description of each module, in the second part of this booklet.

Resits

At the discretion of the Examining Board, candidates who have failed to achieve the marks necessary for the award of an MSc at the end of the taught part of the course may be allowed to resit all or part of the assessment of these modules, once only, on the next occasion that they are offered, in order to reach the standard required either for the award of the diploma or to be allowed to proceed to the dissertation phase of the MSc. The maximum mark which can be obtained when resitting failed modules is 50%.

Resitting the assessment does not give students the right to attend classes for the module.

Projects and Dissertations

Students will be encouraged to start thinking about possible topics for their dissertation after completing 80 credits, i.e. after the first year of the course. Before embarking on the project, a precise description of the work to be undertaken must be submitted and agreed by the course coordinator. Projects based on work carried out for a student's employer are encouraged, provided that they have sufficient academic content.

Each student will be assigned a project supervisor from the academic staff at Aberystwyth. The supervisor's responsibility is to offer guidance to the student both over the work to be carried out and the way the dissertation is to be presented. Communication between students and their supervisors will normally be electronic. Additional assistance will be available from time to time from members of the Aberystwyth staff visiting Singapore.

Students who so wish may carry out their project work in the UK, at Aberystwyth. No additional fees are payable for this and students will have the same rights as other postgraduate students. Travel and subsistence costs will, of course, be the responsibility of the student.

Submission dates

Students must submit their project dissertation no later than 48 calendar months after the date that they started the programme (i.e. the date on which their first module started). If the dissertation is not submitted by the due date, it will be treated as having failed by non-submission and the candidate will be allowed to submit the dissertation on one occasion only, no later than 60 calendar months from the date that they started the programme.

These dates may be extended by prior agreement of the University where over-riding commitments or medical circumstances have delayed the completion of the taught part of the course or of the dissertation. A written case must be presented, supported by independent evidence.

Notwithstanding the formal position described above, students are strongly advised to try to submit their dissertations within about twelve months of completing the taught part of the course.

Resubmissions

If a dissertation fails, the Examining Board may allow it to be resubmitted on one occasion only, no less than six months and no more than twelve months after the date on which the student was informed of the failure. Alternatively, the Examining Board may award a postgraduate diploma.

Note that a distinction cannot be awarded if the dissertation has been resubmitted or has been submitted late.

Entry Requirements

The normal academic entry requirement is a second class honours degree, or better, in a subject other than computing; an equivalent standard is required from graduates of universities that do not use the honours degree system. In addition, applicants are expected to have some general commercial or industrial experience and to be familiar with using personal computers for common applications. (Applicants without such computer experience will be required to enrol on appropriate basic courses offered by Informatics, before starting on the MSc programme proper.)

Applicants over the age of 25 will be considered on an individual basis and, in suitable cases, any one of a wide range of qualifications may be acceptable.

Application should be made through Informatics. Acceptance of applications is subject to approval both by Informatics and the University of Wales, Aberystwyth.

Plagiarism

We have been asked to include the following statement on plagiarism.

UNIVERSITY STATEMENT ON PLAGIARISM

Plagiarism is the act of using someone else's work with an intent to deceive. In academic contexts, the point of the deception is normally to obtain higher marks than you think you would get for your own unaided efforts. There are several ways of going about this. You might decorate your essay with some choice expressions from some other source(s), without making it clear that you have done this. You might take substantial chunks. You might copy from notes or essays written by fellow students or even taken from the Internet. In more extreme cases, students might actually submit work to which they have contributed nothing at all, something that is entirely the work of another mind.

People who do this do it for various motives. A good and ambitious student might do it because s/he desperately wants a very good degree result, and is doubtful if s/he can achieve that on his/her own; or because there is a course in which s/he is relatively weak. A poor student might do it because s/he has been in the pub when s/he ought to have been working and has no work to submit. Sometimes the motives can be very complex. Whatever they are, plagiarism is intellectual dishonesty.

There is of course a very real risk of plagiarism being detected. A student may feel that s/he will get away with downloading material from the Internet and presenting as his/her own work. But it is probably worth noting that if you find it there then the lecturer setting the topic in the first place is also aware of it.

Similarly if you copy a fellow student's work, the chances of it being spotted are very high indeed.

No intellectual endeavour is ever absolutely original. Even the most original minds depend on the thoughts and discoveries of their predecessors. And in most intellectual disciplines, students are expected to demonstrate familiarity with the established literature in their field: indeed, this is one of the key competences that you need to demonstrate in most academic fields. Most of the time, you will be citing articles and books that are especially relevant to your enquiry, and making your own contribution to it. That contribution might not be a great one, especially in the early years of a degree programme; but it will, or should, be your own.

Each Department will have its own subject-specific account of the best ways in which to avoid plagiarism, appearing elsewhere in the Departmental Handbook, and you should familiarise yourself with it.

Sometimes students can be so weak or under-confident in a subject, again especially early on in their studies, that they really find it difficult to tell what is acceptable borrowing from other sources and what is not. Sometimes, unacceptable degrees of borrowing can occur when a student has not actually intended to engage in unfair practice. For this reason, when a member of the academic staff reads work that s/he

suspects is not the unaided work of its supposed author, s/he may not at once notify this to the Chairman of the relevant Examining Board but may discuss it first with the student. University staff will exercise proper academic judgement.

If and when s/he decides to do so, the Chairman will normally interview the student in the presence of the staff member making the enquiry, to establish whether there was an intention to benefit unfairly. The panel may decide that there was not. This, they may then think, is not unfair, *but* bad *practice. They will probably assign an appropriately low mark to the examined element. If, however, the panel is convinced that there is on the face of it a case of* unfair *practice, and if the course element constitutes more than 10 credits' worth of the overall assessment weighting for the year of study, the Chairman will notify the University authorities and what happens next will be governed by the University's* Academic Regulation on Unfair Practice. *The most significant part of this is reproduced in the* Students' Examination Handbook, *which you should possess. If a case of plagiarism is established, the penalties can be very severe indeed and can result in your permanent exclusion from the University.*

Where the assessed element is worth 10 credits or less, departments are authorised to handle the case wholly internally. In most such cases, the mark for the assessed element will be 0 with possibly no opportunity to resit. More severe punishments may also be imposed (e.g. 0 for the module as a whole).

Clearly, however, the most sensible course for a student to pursue, and the course that most students do pursue, is to develop enough academic judgement and self-confidence for them not to be in any danger of such an accusation being made against them. Most students have no wish to gain credit for what they have not themselves contributed, or to gain a qualification that is, even in part, a bogus achievement.

As you see, it is important to indicate clearly in your own work where you have included the work of others. In Computer Science this could include reuse of designs and programs as well as copying or quoting text. Make sure you understand how to acknowledge the work of others in all your submissions. Ignorance of how to do this is not a valid defence.

The following simple guidelines are intended to help you avoid to avoid straying from legitimate and desirable co-operation into the area of plagiarism:

- append a bibliography to your work listing all the sources you have used, including electronic sources;
- surround all direct quotations with inverted commas, and cite the precise source (including page numbers, or the URL and the date you accessed it if the source is on the Web) either in a footnote or in parentheses directly after the quotation;
- use quotations sparingly and make sure that the bulk of the work is in your own words;
- remember that it is your own input that gives a piece of work merit. Whatever sources you have used, the structure and presentation of the argument should be your own. If you are using electronic sources, don't cut and paste sections into your work. If you are using books or papers, put them aside when you actually sit down to write. In this way you won't be tempted to copy in material that you don't understand, or be at risk of unintentionally copying in more material than a brief quotation, or of accidentally leaving quotations unmarked.

Do keep a sense of proportion, and exercise common sense and judgement. For example, it is not necessary to attribute to a source statements which have passed into the public domain and become commonplace. It is usually unnecessary to attribute lecture material, though again you should avoid quoting copiously, and you should not rely wholly on lecture notes.

Books and Equipment

According to their publishers, all the books included in the reading lists for the modules are available in Singapore. In practice, it is not always easy to obtain them quickly and it is advisable to order them well in advance. In cases of difficulty, the book stores that sell over the Internet may be the most convenient way of obtaining the books. It is advisable to look at both UK and US sources - in some instances UK sources may be significantly cheaper.

You will need ready access to a reasonably powerful PC running an up-to-date version of Microsoft Windows and Office. Other software will be provided. If you do not already have access to e-mail, you can arrange this through Informatics. e-mail is our normal means of communicating on a regular basis with our students.

Academic Support and Assistance

While students who want advice or assistance with academic matters relating to the course or who want to comment on any aspect of the course are encouraged to approach any of the Aberystwyth lecturers, the formal point of contact is through the course co-ordinator who can be contacted via the department. The address of the Computer Science Department in Aberystwyth is:

Department of Computer Science, UWA, Penglais, Aberystwyth, Ceredigion, SY23 3DB, UK

Tel: +44 1970 622 424 Fax: +44 1970 622455

http://www.aber.ac.uk/compsci/

cs-mscsingadmin@aber.ac.uk

Student Feedback

The department tries hard to keep the quality of its courses as high as possible. In order to do this it looks for input to employers, to professional institutions, to colleagues in other universities and, most importantly, to its students. We seek your views in several ways, as described below, and we always welcome informal comment. The courses, in their current form, have benefited from student input over the years; please play your part in making them better for future generations of students. Remember, however, that we often get conflicting comments - employers, professional institutions and students do not always agree with each other.

Formal sessions are held, during modules run by senior members of staff, where your opinions on individual modules and the scheme as a whole will be sought. These sessions will be particularly relevant to possible strategic changes.

Questionnaires will be used to collect more specific feedback.

Immediate or longer term feedback on elements of particular modules are welcomed by the members of staff delivering them. This can be provided during delivery of the module or via e-mail afterwards.

We also need your help if problems arise with equipment or with administration. If we know about a problem, there is a good chance that we can solve it quickly; if we don’t know about it, there’s nothing we can do.

Supplementary Charges

In addition to the course fees, additional charges may be levied as follows:

Sitting or resitting an examination for a module at a time other than that at which an examination for the module is scheduled	£100
Resitting an examination for a module at a time when an examination for the module is scheduled	£50
Retaking a complete module, including classes, course work and examination	£250

MFB and MJM, August 2001

Module Identifier	**CO11020**		
Module Title	**INTRODUCTION TO PROGRAMMING**		
Academic Year	**2002**		
Coordinator	**Dr Mora McCallum**		
Semester	**Available All Semesters**		
Other staff			
Pre-requisite	**Available only to students taking the Diploma/MSc in Computer Science scheme in Singapore.**		
Course delivery	**CONTACT HOURS**	**55 hours of contact time; lectures, practicals, workshops. 150 hours of private study, practical work and assesment.**	
Assessment	**Assignment**	**(A1) Three practical assignments**	**75%**
	Assignment	**(A2) Final written assignment**	**25%**
	Supplementary examination	**There is no provision for supplementary examinations or resits.**	

Brief description

There is much more to computing than programming and many graduates from the Diploma/MSc course may never need to do any programming in their professional careers. Nevertheless, an understanding of programming and, more generally, of the software development process is an important part of the education of anyone who wishes to be an IT professional. Such an understanding needs some practical skill and experience and this is what this module provides.

Aims

To make students understand what is involved in software development and to give them the basic skills necessary to develop well-structured, non-trivial programs in a well-designed programming language using a modern environment.

Learning outcomes

On successful completion of the module, students should:

* be able to develop non-trivial Java programs to operate in the environment they have studied (A1);
* demonstrate an understanding of the nature and need for testing by being able to test the programs they have written (A1);
* have a mental model of a computer, adequate to understand what is involved in developing programs (A1, A2);
* understand the concept of an algorithm demonstrated through an ablility to design simple algorithms (A1, A2);
* demonstrate how software components are combined to form complete systems (A1, A2);
* demonstrate an understanding of the idea of the software life cycle and the stages within it (A1).

Syllabus

1. INTRODUCTION TO COMPUTING AND ALGORITHMS
Introduction to the basic computer organisation and environment that will be used for the course. The idea of an algorithm, abstraction, and programs. The software development life cycle.

2. THE ELEMENTS OF A SIMPLE PROGRAM
Introduction to Java. Types, variables, statements. Branches and loops. Arrays.

3. OBJECT-ORIENTED PROGRAMMING
Introduction to objects and classes. Elementary design of object-oriented systems. Use of standard notation for expressing designs.

4. PROGRAMMING IN THE LARGE
Object-oriented programming in Java. Classes in Java. Inheritance. Information hiding. Robust programming, exceptions. Component libraries and their use.

5. PROGRAM TESTING
Techniques and aids for error detection.

6. PERSISTENT DATA

Input/output and files. File handling in Java.

7. PRACTICAL WORK
In class practical work and assignments.

Reading List

Ivor Horton. (March 1999) Beginning Java 2. Wrox Press Inc.

J. M. Bishop. (2001) Java Gently: Programming Principles Explained. 3rd edition. Addison-Wesley Pub Co.

S. Heller. (1998) Who's Afraid of Java. AP Professional.

Y. Daniel Liang. (2000) Introduction to Java Programming. 3rd Edition. Prentice Hall.

Walter Savitch. (2000) Java: An Introduction to Computer Science & Programming. 2nd edition. Prentice Hall.

Elliot B. Koffman and Ursula Wolz. (Aug 1998) Problem Solving with Java. Addison-Wesley.

Samuel N. Kamin, M. Dennis Mickunas, and Edward M. Reingold. (Nov 1997) An Introduction to Computer Science: Using Java. WCB/McGraw-Hill.
Cay Horst Mann. (2000) Computing Concepts with Java 2 Essentials. John Wiley.

John Lewis and William Loftus. (2000) Java Software Solutions. Addison Wesley.

Patrick Henry Winston, Sundar Narasimhan. (2001) On to Java. 3rd edition. Addison-Wesley Pub Co.

Stephen J. Chapman. (1999) Java for Engineers and Scientists
. Prentice Hall.
It is considered essential that students buy one of these general texts on Java. Exactly which is left to your own personal preference. Advice will be offered in lectures..

Module Identifier	**CO21120**		
Module Title	**DATA STRUCTURES, ALGORITHMS AND SOFTWARE DESIGN**		
Academic Year	**2002**		
Coordinator	**Dr Mora McCallum**		
Semester	**Available All Semesters**		
Other staff			
Pre-requisite	**CO11020**		
Pre-requisite	**Available only to students taking the Diploma/MSc in Computer Science scheme in Singapore.**		
Course delivery	**CONTACT HOURS**	**55 hours of contact time; lectures, practicals, workshops. 150 hours of private study, practical work and assesment.**	
Assessment	**Exam**	**(A1) 2 Hours**	**75%**
	Assignment	**(A2)**	**25%**
	Supplementary examination	**Supplementary examination will take the same form, under the terms of the Department's policy.**	

Brief description

In the 40 or so years that people have been developing software, a range of common tasks, such as sorting and searching, have been identified and a body of knowledge has been accumulated about how these tasks are best carried out. This knowledge usually consists of various ways of structuring the data, a range of different algorithms, and performance analysis that enables a designer to chose the algorithm and data structure most appropriate to the circumstances of the system, and to predict how the system will perform.

This module introduces students to this body of knowledge and sets it into the context of modern software design and structuring techniques.

Aims

The aims of this module are to:

* introduce students to the standard data structures and algorithms that form the common currency of software design;
* develop students' understanding of the object-oriented model of software;
* introduce students to the analysis and prediction of performance.

Learning outcomes

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

* demonstrate their understanding of the principles of abstraction and encapsulation as they apply to the design of abstract data types and programs (A1, A2);
* analyse and evaluate the time and space behaviour of algorithms and understand how this is expressed and determined (A1, A2);
* recognise the importance of this analysis in the design of software (A1, A2);
* recognise the importance of the classes P and NP in the analysis of algorithms (A1);
* describe some of the main approaches to algorithm design such as greedy algorithms, divide and conquer and dynamic programming (A1);
* demonstrate judgement in evaluating and choosing appropriate data structures and algorithms for a range of programming problems (A1, A2);
* design and implement significant programs in Java (A2).

Syllabus

1. Introduction - 15 Lectures

Course Overview. An introduction into time/space complexity. Mathematical underpinnings. Issues of correctness as they relate to the definition of ADTs. The key ideas of abstraction and encapsulation. Notations for describing ADTs. Java support for their implementation: packages, exceptions and interfaces.

2. Introduction to Complexity - 3 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.

3. Classes of Algorithm - 4 Lectures
An overview will be given on the different classes of algorithm; for example, divide and conquer and greedy algorithms. Genetic algorithms will also be discussed. P and NP.

4. Recursion - 3 Lectures
An introduction to recursive thinking. Examples of recursion.

5. Storing and Retrieving Data by Key (1) - 12 Lectures
This problem will be used to motivate the discussion of a wide variety of 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. Linked representations; an introduction to hashing, binary search trees and heaps.

6. Storing and Retrieving Data by Key (2): External storage - 4 Lectures
Performance issues. Hashing and B-tree organisations. The Hashable class in Java.

7. Representing Text - 4 Lectures
String matching algorithms and their performance. Search engines case study.

8. Sorting - 4 Lectures
A comparison of divide and conquer, priority queue and address calculation based sorting algorithms. Performance characteristics of these algorithms will be discussed.

9. Representing Complex Relationships: Graphs - 6 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).

Reading List

T. Budd. (2001) Classic Data Structures in Java. Addison-Wesley Pub Co.

Michael Main. (Oct 1998) Data Structures and Other Objects Using Java. Addison-Wesley.

T.A. Standish. (1998) Data Structures in Java. Addison Wesley.

Alfred Aho, John Hopcroft, and Jeffrey Ullman. (1983) Data Structures and Algorithms. Addison Wesley.

Alfred Aho and Jeffrey Ullman. (1995) Foundations of Computer Science. W H Freeman & Co..

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

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

Module Identifier	**CO23010**		
Module Title	**COMPUTER ARCHITECTURE**		
Academic Year	**2002**		
Coordinator	**Dr Mora McCallum**		
Semester	**Available All Semesters**		
Other staff			
Pre-requisite	**CO11020**		
Pre-requisite	**Available only to students taking the Diploma/MSc in Computer Science scheme in Singapore.**		
Course delivery	**CONTACT HOURS**	**34 hours of contact time plus about 65 hours of self study, practicals and assessment.**	
Assessment	**Exam**	**(A1) 2 Hours**	**80%**
	Course work	**(A2)**	**20%**
	Supplementary examination	**Supplementary examination will take the same form, under the terms of the Department's policy**	

Brief description

There is a relationship between software design, hardware design and the performance of a system as a whole. Those who build software systems need some understanding of this relationship, at the level of principles rather than fine detail. No prior knowledge of the subject area is required; the course begins with the basic ideas of number systems, simple electrical laws, logic functions and their electrical equivalents. The course then looks at how more complex devices can be created from these primitive building blocks. Having arrived at the principal components of a microprocessor based system, we look at the way these components interact and their roles in the execution of simple programs. The laboratory work associated with the course is designed to consolidate the lecture material using PC-based training tools.

Aims

Students successfully completing this module will have a broad functional understanding of computer architecture, an awareness of the hardware software interface, an understanding of the trade-offs between hardware and software, and an understanding of the factors that affect system performance.

Learning outcomes

On successful completion of this module, a student should be able to:

* analyse a block diagram of a computer and explain how it works at the level of logic gates (A1);
* analyse and develop low level programs and describe how they are executed by a CPU (A1, A2);
* describe how a computer performs input and output operations (A1);
* explain how abstract concepts in high-level languages, such as `function call' or `local variable', are implemented in machine code (A1);
* judge the applicability of high and low level language programming (A1).

Syllabus

1. What is a computer? - 4 Lectures
Block diagram overview; CPU, memory, I/O, Bus. Memory, Digital Logic; pigeon-hole model, address and contents, bits bytes and words.

2. Buses - 2 Lectures
Address, data and control buses. Basic data transfer.

3. Inside the CPU - 3 Lectures
Simple examples of instructions. The fetch-execute cycle and the program counter. Registers. ALU. Control unit. Implementing a machine code in hardware. Digital logic.

4. A real CPU example: Motorola 68000 and 68HC11 - 4 Lectures
Some machine codes and mnemonics. Addressing modes. Assembly code.

5. Executing high-level software - 4 Lectures
Machine-code equivalents of high-level constructs. Function calls. Stack frames and local variables.

6. I/O - 5 Lectures
Reading and writing data. Interrupts. Transfering large amounts of data; DMA, block I/O.

7. Exercises - 4 Practicals
Use a CPU simulator to watch instruction execution. Assembly language comprehension (probably, but not necessarily, by writing a program).

Reading List

Ronald J. Tocci and Frank J. Ambrosio. (2000) Microprocessors and Microcomputers: Hardware and software.. 5th. Prentice Hall. (Recommended text)

Module Identifier	**COM1220**		
Module Title	**THE SOFTWARE DEVELOPMENT LIFE CYCLE**		
Academic Year	**2002**		
Coordinator	**Dr Mora McCallum**		
Semester	**Available All Semesters**		
Other staff			
Pre-requisite	**CO11020**		
Pre-requisite	**CO21120**		
Pre-requisite	**Available only to students taking the Diploma/MSc in Computer Science scheme in Singapore.**		
Course delivery	**CONTACT HOURS**	**55 hours of contact time; lectures, practicals, workshops. 150 hours of private study, practical work and assesment.**	
Assessment	**Exam**	**(A2) 2 Hours**	**40%**
	Group project	**(A1)**	**50%**
	Attendance and participation	**(A3) 1 presentation plus contributions**	**10%**
	Supplementary examination	**No supplementary or resit examination available.**	

Aims

This module aims to expose students to best practice in software engineering and to develop professional skills enabling students to be a valuable part of a software development team. Specifically, it aims to enable students to:

* select best practices in the engineering activities of project management, quality assurance and standards compliance;
* identify and employ appropriate practices for the specification, design, testing and operation of large software systems;
* involve students in the development of a piece of software which approximates as closely as possible in the university environment to the software development conditions found in industry.

Learning outcomes

The major learning outcome of this module is that the student should:
1. be able to employ best professional software engineering practices in order to complete a medium-sized software project to current industrial standards. (A1)

In addition, on successful completion of this module, students should be able to:
2. demonstrate mastery of advanced concepts in software engineering (A2, A3)
3. select appropriate advanced software engineering techniques to apply to challenging industrial problems (A2, A3)
4. recognise potential problems in software projects, and be able to intervene to avoid them (A1, A2)

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 - 3 Lectures; 3 Seminars
Description of the phases of a range of software life cycles (including the Waterfall, Prototyping, RAD and Spiral models) and the major deliverables and activities associated with each phase. Rapid application development. Personal software metrics. Extreme programming. Software process improvement.

3.Project Management - 2 Lectures, 2 Seminars
Planning and cost estimation. Progress monitoring. Team structure and team management. Project management in industry.

4.Quality Management - 2 Lectures, 1 Seminar
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:

6.Requirements Engineering - 3 Lectures, 2 Seminars
The IEEE standard for requirements specifications. Validation of requirements by e.g., prototyping. Deficiencies in the traditional approach to requirements. Use of UML in requirements gathering. Advances in requirements engineering.

7. Design - 3 Lectures, 2 Seminars
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. State diagrams. Relevant UML notations: packages, sequence and activity diagrams, active objects. User interface design.

8.Implementation and maintenance - 2 Lectures
Choice of language. Cutover. Types of maintenance. Maintenance process. Refactoring.

9.Testing - 2 Lectures
Testing strategies. Testing tools: static and dynamic analysers, test harnesses and test data generators, simulators. Performance testing. Regression testing. User documentation and training.

10.Tutorials
A tutorial meeting will be associated with this course. The tutorial will be used to organise group project activities.

11. Seminars
A list of papers on advanced software engineering topics will be distributed, and the papers will be presented and applied to sections of the syllabus during seminars, as indicated above.

Reading List

Ian Sommerville. (2001) Software Engineering. 6th Edition. Addison Wesley. (Recommended text)

Roger S. Pressman. (2000) Software Engineering: A practitioner's approach. 5th Ed.. McGraw-Hill. (Recommended text)

Module Identifier	**COM3110**		
Module Title	**THE UNIX ENVIRONMENT AND THE C LANGUAGE**		
Academic Year	**2002**		
Coordinator	**Dr Mora McCallum**		
Semester	**Available All Semesters**		
Other staff			
Pre-requisite	**CO21120**		
Pre-requisite	**Available only to students taking the Diploma/MSc in Computer Science scheme in Singapore.**		
Course delivery	**CONTACT HOURS**	**34 hours of contact time plus about 65 hours of self study, practicals and assessment.**	
Assessment	**Exam**	**2 Hours**	**100%**
	Supplementary examination	**Supplementary examination will take the same form, under the terms of the Departments policy.**	

Brief description

This module is designed to to provide experience of using the Unix environment and supporting tools as well as an understanding of the programming language C.

Learning outcomes

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

* demonstrate a good understanding of the nature of the computer language "C" including the more challenging aspects of the language;
* apply the facilities of the language "C" to technically advanced problems;
* describe the differences between object oriented languages (such as Java) and non-OO languages (such as C) and make appropriate choices between such languages to solve a range of realistic problems;
* demonstrate an understanding of the structure and paradigms of the Unix operating system;
* demonstrate a mastery of the use of a selection of Unix tools and make appropriate decisions on the choice between the use of existing Unix tools or the development of new software systems for realistic applications.

Syllabus

1. Introduction
Overall introduction to the module.

2. Unix at the command line
An introduction to the alternative Unix shells. Shell built-in commands and commonly used external commands and editors.

3. Shell Script programming
The programming language provided by a selected Unix shell in common usage.

4. Tools of the Unix Environment
Purpose and usage of Unix environment tools such as sed, sort, uniq, awk, grep and so on.

5. Basic Concepts of "C"
History of the C language, philosophical differences between C language design and Java. Basic form of a C program compared with that of a Java program. Using the compiler.

6. Control Structures
Sequence, branching and iteration in C compared with that of Java.

7. Basic Data Structures
Review of basic data types and operators in C.

8. Functions
Discussion of ways in which functions are implemented, and used in C, including parameter passing mechanisms. Input/Output.

9. Composite Data Structures
A first discussion of Arrays in C.

10. Software Support Tools
Make, Lint, Debuggers. Libraries and library utilities.

11. C Programming Style and Portability
Language standards. Portability. Programming standards.

12. Arrays, Pointers and Functions
A discussion of pointer data types, how they relate to arrays, and how they contrast with references to Java objects.

13. Dynamic Data Structures
Implementation of various record structures and dynamic structures. Pointers. Malloc. Examples in C. Parallels will be drawn with how the internals of Java do this for you.

14. Pitfalls
Major problem areas. Design rationale of C and of Java in problem areas.

15. Further Features
C preprocessor, header files, conditional inclusion, macro substitution, bitwise operators, casts, enumeration, scope, static and external declarations, separate compilation.

Reading List

A. Kelley and I. Pohl. (1998) A Book on C: programming in C. 4th Edition. Addison-Wesley Pub Co. (Recommended text)

Ellen Siever (Editor), Jessica P. Hekman, Stephen Figgins and Stephen Spainhour. (2000) LINUX in A Nutshell: A Desktop Quick Reference. O'Reilly & Associates. (Consult for futher information)

Module Identifier	**COM5010**		
Module Title	**INTRODUCTION TO COMMUNICATIONS AND TELEMATICS**		
Academic Year	**2002**		
Coordinator	**Dr Mora McCallum**		
Semester	**Available All Semesters**		
Other staff			
Pre-requisite	**CO11020**		
Pre-requisite	**Available only to students taking the Diploma/MSc in Computer Science scheme in Singapore.**		
Course delivery	**CONTACT HOURS**	**34 hours of contact time plus about 65 hours of self study, practicals and assessment.**	
Assessment	**Exam**	**2 Hours**	**100%**
	Supplementary examination	**Supplementary examination will take the same form, under the terms of the Department's policy.**	

Brief description
The purpose of this module is to present an introduction to the problems encountered and the methods used in modern computer based communications systems.

Aims
This module aims to give an introduction to the area of Computer Communications and Telematics. The two main aims of the module are to:

* provide sufficient foundation for students to study COM5610 ;

* provide sufficient experience of the subject matter to allow students to use the knowledge in their careers.

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

* plan networks that are cost effective and realistic in terms of products and services currently available;
* critically assess proposed networking solutions;
* investigate, and propose solutions to, problems of quality of service;
* assess the effect of likely technological and regulatory developments on existing network applications.

Syllabus
1. Introduction

2. History of Communications
A brief history of the development of both the technology and regulation of communication systems. Common Carriers; UK carriers: British Telecom, Mercury, Kingston and the new market entrants. Internet and its history and evolution. The UK academic network and its development, regulation and operation.

3. Telematic Applications
The range of applications, their characteristics, requirements and usage.

4. Basics of Data Communication
Analogue and digital data transmission; Synchronous and Asynchronous transmission; Parallel and Serial transmission; Modems and the PSTN, concentrators, multiplexors; Co-ax, twisted pair, fibre optic media; Speed, distance, error rates of various transmission media.

5. Local Area Networks
Bus, Ring, Star topologies; Cost of attaching devices to networks; Media access and sharing strategies.

6. Wide Area Network technologies and services

Public Switched Networks and private lines; Kilostream, Megastream and similar services; N-ISDN; Examples of WANs.

7. Standards
The needs for standards and the area they cover; Standards setting process; Standards setting bodies; BSI, ISO, ANSI, IEEE, CCITT, IAB; IETF; De facto standards.

8. An Introduction to Internet and Other protocols
ISO Model, IP, TCP, UDP, FTP, Telnet.

9. Security of Information Systems
Need for security and its cost; Risk assessment; Simple techniques; Passwords and Badge readers; Deficiencies of simple approaches; The Orange Book standard.

Reading List

Gordon Brebner. (1997) Computers in Communication. McGraw Hill. (Recommended text)

Lewis Mackenzie. (1998) Communications and Networks. McGraw Hill. (Recommended text)

Module Identifier	**COM5610**		
Module Title	**THE INTERNET: TELEMATIC APPLICATION DESIGN AND CONSTRUCTION**		
Academic Year	**2002**		
Coordinator	**Dr Mora McCallum**		
Semester	**Available All Semesters**		
Other staff			
Pre-requisite	**COM5010**		
Pre-requisite	**Available only to students taking the Diploma/MSc in Computer Science scheme in Singapore.**		
Co-requisite	**CO21120**		
Course delivery	**CONTACT HOURS**	**34 hours of contact time plus about 65 hours of self study, practicals and assessment.**	
Assessment	**Exam**	**2 Hours**	**100%**
	Supplementary examination	**Supplementary examination will take the same form, under the terms of the Department's policy.**	

Brief description

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

Aims

This module is designed to provide an insight into the nature and potential of Telematics Applications together with an understanding of the component technologies. In particular we will look at how the Internet operates and the basis it provides for telematic systems support.

Learning outcomes

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

* demonstrate a critical understanding of a range of issues associated with the design of telematic applications;
* describe a range of contrasting facilities for the design and construction of distributed applications and assess their relative applicability to real world problems;
* express a mastery of the particular issues associated with the handling of multimedia traffic
* show an understanding of the need for secure communications, describe various key technologies used to provide it and assess their relative applicability to real world problems.

Syllabus

1. Telematics Systems Creation for the Internet
Introduction and multifunctional workstations.

2. Java's Support for Internet Communications
Java RMI, Java socket access.

3. Java Applet Construction
Applet construction and related issues of the Java security model.

4. Network based Multimedia Applications
Issues in audio/video application construction, protocols such as RTP and RTSP and the Java media classes such as JMF, Sound API, Speech API, Telephony API.

5. Secure Communications
Authentication, key management and cryptosystems.

Reading List

Reading List

Students will be informed of appropriate up-to-date material..

Module Identifier	**COM6010**		
Module Title	**ARTIFICIAL INTELLIGENCE**		
Academic Year	**2002**		
Coordinator	**Dr Mora McCallum**		
Semester	**Available All Semesters**		
Other staff			
Pre-requisite	**CO21120**		
Pre-requisite	**Available only to students taking the Diploma/MSc in Computer Science scheme in Singapore.**		
Course delivery	**CONTACT HOURS**	**34 hours of contact time plus about 65 hours of self study, practicals and assessment.**	
Assessment	**Supplementary examination**	**Supplementary examination will take the same form, under the terms of the Department's policy.**	
	Exam	**2 Hours**	**100%**

Brief description

This module concentrates on the technology and methodology of artificial intelligence (AI) practice. The aim is to show how practical knowledge-based systems can be implemented, developed and evaluated.

Aims

This course aims to give students a good understanding of a variety of AI systems, from expert systems through machine learning to adaptive computing. This will enable them to appreciate what current AI systems can and cannot do, and the circumstances in which their use is appropriate.

Learning outcomes

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

* identify the main application areas in which artificial intelligence has been applied;
* discuss some of the main controversies within and around artificial intelligence, particularly those concerning the nature of intelligence, and the limits of artificial intelligence;
* explain the contributions of artificial intelligence to computing in general, and to industry and commerce, particularly in the area of expert systems;
* describe and assess artificial intelligence techniques for a selected set of application topics;
* make intelligent use of artificial intelligence software.

Syllabus

1. Introduction to AI
What is AI? Introduction to the range of applications of AI. Why do we need AI and how do we use it?

2. Intelligent Agents
How can we build an intelligent robot? Sensing, Action and Cognition. The symbolic approach. Memory, representation and reasoning.

3. Machine Vision
How can robots see? The nature of the vision task. Computer vision and image processing.

4. Knowledge
How can robots think? Knowledge representation methods. Reasoning and inference.

5. Expert Systems
How can human expertise be automated? Example applications and commercial successes. How to build an expert system - system concepts and architectures. Rule-based systems: design, operation and worked examples. Knowledge bases and knowledge based systems.

6. Artificial Brains

How can robot brains be built? Artificial neural nets, pattern recognition and learning.

7. Search and reasoning
Why do we need search? Evaluation of search strategies. Un-informed search techniques. Informed search techniques.

8. Machine Learning
How can a computer program learn? Inductive learning. Structural methods. Genetic algorithms.

9. Learning application
Case-based reasoning. Data mining.

Reading List

S.J. Russell and P. Norvig. (1995) AI: A Modern Approach. Prentice-Hall. (Recommended text)

Alison Cawsey. (1998) The Essence of Artificial Intelligence. Essence of Computing Series. Prentice Hall. (Recommended text)

J. Giarratano and G. Riley. (1998) Expert Systems: Principles and programming. Boston PWS. (Consult for futher information)

E. Rich and K. Knight. (1991) Artificial Intelligence. 2nd. McGraw Hill. (Consult for futher information)

Module Identifier	**COM7020**		
Module Title	**DATABASES AND DATA ANALYSIS**		
Academic Year	**2002**		
Coordinator	**Dr Mora McCallum**		
Semester	**Available All Semesters**		
Other staff			
Pre-requisite	**Available only to students taking the Diploma/MSc in Computer Science scheme in Singapore.**		
Course delivery	**CONTACT HOURS**	**55 hours of contact time; lectures, practicals, workshops. 150 hours of private study, practical work and assesment.**	
Assessment	**Exam**	**3 Hours**	**100%**
	Supplementary examination	**Supplementary examination will take the same form, under the terms of the Department's policy.**	

Brief description

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

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.

Learning outcomes

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

* design and validate a database from an enterprise description involving ambiguity and uncertainty;
* use technical judgement to improve an initial logical design and choose an appropriate physical design to enhance run time performance;
* implement a database design and access the database through an appropriate programmatic interface;
* describe the facilities of modern database management systems and justify the choice of a particular system on limited technical grounds;
* explain and provide a rationale for relational, object and object-relational database concepts;
* understand and use a data model, in particular, the relational data model and contrast this with an object model;
* plan, design and manage the improvement and extension a database design.

Syllabus

1. Database Systems Concepts
The difference between databases and files. Databases, DBMS and applications programs. Why databases are needed. The idea of a data model. Available models.

2. Relational Algebra
Definition of a relation. Standard relational operators. Referential integrity.

3. Normalisation
Functional dependencies. Normalisation: first to fifth normal forms, domain/key normal form. Bottom up analysis.

4. Relational Modelling
Top down analysis. Enterprise modelling. Entities and relationships. Connection traps. The design of relations. Transformation of an E-R model into a relational schema.

5. Implementing a Database
Overview of the facilities provided by Access. Queries, queries as views. Built-in functions. Forms and reports. Event handling.

6. SQL
Outline of the language. The language as an implementation of the relational model. DDL as a contrast to Access facilities. Nested queries and sub-queries.

7. Missing values
The need for nulls. Theoretical and practical problems. Null values and the outer join.

8. Application programs
Procedural interfaces. SQL in applications programs. The data dictionary. General integrity constraints: DBMS facilities versus application code. Interoperability of database systems. Back up and recovery.

9. Physical Database Design
Table design. Enterprise rule design. Transactional analysis and index choices. Controlled redundancy.

9. Database Lifecycle
Synthesis and revision: logical design, physical design, monitoring and tuning.

9. Distributed Databases, Concurrency and Transactions
Introduction to concurrency. What is a distributed database, why should one wish to use one, and what problems will it bring? Transaction processing. Backup and recovery.

10. Older Data Models
Hierarchical and network models: how they relate to the relational model.

11. Introduction to Object Databases
Perceived weaknesses of the relational model. Possible benefits of an object model. ODMG model.

12. Building and Manipulating Object Databases
Design; use of the UML. ODL and database aspects of CORBA. OQL and OML.

13. Object Relational Systems
Comparisons of relational and object systems. Object oriented extensions to relational systems. SQL3.

Reading List

Thomas Connolly and Carolyn Begg. (2001) Database Systems: A practical Approach to Design, Implementation and Management. 3rd if out. Addison-Wesley. (Recommended text)
Thomas M Connolly and Carolyn E. Begg. (2000) Database Solutions: A step-by-step approach to building databases.. 1. Addison -Wesley. (Recommended text)
Students should await the latest publication information before deciding on a text.

Module Identifier	**COM8110**		
Module Title	**SOFTWARE MANAGEMENT AND THE SOFTWARE INDUSTRY**		
Academic Year	**2002**		
Coordinator	**Dr Mora McCallum**		
Semester	**Available All Semesters**		
Other staff			
Mutually exclusive	**Only available to students following the Dip/MSc in Computer Science in Singapore**		
Course delivery	**CONTACT HOURS**	**34 hours of contact time plus about 65 hours of self study, practicals and assessment.**	
Assessment	**Report**	**(A1) A group assessment involving the preparation of an invitation to tender or a response. (Due to this form of assessment, no supplementary or resit examination is available).**	**50%**
	Essay	**(A2)**	**50%**

Brief description

The software industry is now one of the largest and most complex in the world. The individual players within it include companies of such size and complexity that they present unique problems of management. A software professional who intends to rise to a senior management position in the industry must at least be aware of the its structure and characteristics and of its management problems. The purpose of this module is to provide an introduction to management theory specifically in the context of the software industry and an initial framework for the industry structure that allow the new professional to make sense of the many different types of company that he or she will encounter.

Learning outcomes

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

* select appropriate procurement strategies, including contractual arrangements, and identify appropriate tenderers for substantial software procurements (A1);
* participate at a professional level in the preparation of invitations to tender and responses to such invitations (A1);
* critically assess the human resource strategy of a software company (A2);
* assess the effects of legislation relating to the engineering profession and professional codes of conduct, as they exist in different countries, on the operations of a software company (A2).

Syllabus

1. The Industry

The nature and characteristics of the software industry: broad and narrow definitions. Classification of the products of the industry. Treatment of software assets under different accounting regimes.
Structure of the software industry: distribution by size, ownership, specialisation. The growth of outsourcing and its effect on the structure of the industry in different countries. Treatment of software in the calculation of GDP.

2. Procurement

Bespoke software v. packaged software. Identifying potential suppliers.
Procurement strategies: study of a range of strategies used for procuring large systems by governments in different countries. Case studies of some major procurement failures. Problems occasioned by the need for long-term maintenance of large software systems.
Contracts for the provision of bespoke software: fixed price, time and materials. Contracts for packaged software. Use of standard terms and conditions. The Unfair Contract Terms Act 1977.

3. Management

Mission statements, aims and objectives. The need for strategic planning and the problems of doing it in technology-driven industry. Application pull v. technology push. Comparison of the problems of strategic planning in hi-tech products companies and service companies.
Special problems of human resource management in the software industry: difficulties caused by a project-based environment; need to keep technical knowledge up to date; effect of strong competition for qualified staff. Motivational theory: application of theories such as Maslow's hierarchy of needs and Herzberg's two factor theory to the software industry.
Case study of a medium-sized bespoke software company.

4. Regulation
Regulation of the engineering profession in the UK, the USA and continental Europe. The Washington Accord and the Bologna Declaration. Codes of conduct: the BCS code, the IEEE-CS/ACM joint code.

Reading List

M.F. Bott, J.A. Coleman, J. Eaton, and D. Rowland. (2001) Professional Issues in Software Engineering. 3rd Ed.. Taylor and Francis, London. (Recommended text)
There is no single text that covers all the material in this course and, indeed, much of the material is not available in book form at all. The recommended text covers a significant part of the course and contains many useful references, including the addresses of relevant web sites:.
Students will be expected to consult web sites, government reports, and examples of commercial documents..

Module Identifier	**COM9060**
Module Title	**MSC PROJECT**
Academic Year	**2002**
Coordinator	**Dr Mora McCallum**
Semester	**Available All Semesters**
Other staff	
Pre-requisite	**Successful completion of all other modules; see regulations.**
Mutually exclusive	**Only available to students registered for the MSc in Computer Science in Singapore.**
Course delivery	
Assessment	**Dissertation** **100%**

Brief description

The purpose of the project, on which the dissertation is based, is to demonstrate that the student has acquired the ability to undertake and satisfactorily complete a major piece of work in computing.

Aims

The aim of the MSc project is to demonstrate that students can:

* bring together the knowledge acquired from the various modules in the programme and apply it to a major task;
* use the professional and academic literature to extend their knowledge to meet the challenges of the project;
* critically evaluate other people's work and their own.

Learning outcomes

On completion of the project, students will have:

* carried out a comparatively small scale piece of computing work, independently;
* been informed in their approach to this work by the available literature;
* thought critically about their work;
* analysed and described the work in a clear and well written dissertation.

Reading List

Frank Bott. (1999) MSc in Computer Science: Guidelines for Projects and Dissertations. Computer Science Dept, UWA. (Recommended text)
J A Sharp and K Howard. (1996) The Management of a Student Research Project. 2nd. Gower,Aldershot. (Recommended background)