Faculty of
Computing, Engineering
and Mathematical Sciences

BRISTOL

Faculty of Computing, Engineering and Mathematical Sciences

BSc (Hons) Computer Science

Version 2

Definitive Documentation – November 2007

BSc Computer Science

Contents Page

Programme Specification

Section 1: Basic Data

Awarding institution/body	UWE
Teaching institution	UWE
Faculty responsible for programme	Computing, Engineering and Mathematical Sciences
Programme accredited by	N/A
Highest award title	BSc (Hons) Computer Science
Default award title	BSc Computer Science
Interim award title	Dip HE Computer Science Cert HE Computer Science
Modular Scheme title (if different)	MAR
UCAS code (or other coding system if relevant)	G400
Relevant QAA subject benchmarking group(s)	Computing
On-going/valid until* (*delete as appropriate/insert end date)	
Valid from (insert date if appropriate)	September 2003

Authorised by… **Date:…**

Version Code

For coding purposes, a numerical sequence (1, 2, 3 etc.) should be used for successive programme specifications where 2 replaces 1, and where there are no concurrent specifications. A sequential decimal numbering (1.1; 1.2, 2.1; 2.2 etc) should be used where there are different and concurrent programme specifications

Section 2: Educational Aims of the Programme

The BSc in Computer Science has the following general aims:

1. To prepare students for entry into the computing profession and the more general challenges of professional and personal life.

2. To inculcate in students problem-solving and other transferable skills that will be valuable to them in any career.

3. To prepare students for progression to study for higher degrees in Computer Science.

4. To continue the development of those general study skills that will enable students to become independent, lifelong learners.

The BSc in Computer Science has the following specific aims:

1. To provide a coherent and broad based coverage of the theory of computer science and its application to practical problems.

2. To enable students to appreciate the problems that can arise in computer science and to provide them with the appropriate skills to select and apply appropriate methods and technologies to solve them.

3. To encourage students to uphold professional, ethical and social standards and to keep up to date with recent technological and theoretical developments.

4. To provide exposure to the body of research that underlies the use of computers and to develop familiarity with some major themes within it.

5. To enable a student to work in any area deemed to be the subject of research into computer science, such as AI, expert systems, machine intelligence, compiler design.

6. To develop the students' understanding of the importance of solving complex ill-defined problems in any domain, though with particular reference to the development of software.

Section 3: Learning Outcomes of the Programme

The award route provides opportunities for students to develop and demonstrate knowledge and understanding, qualities, skills and other attributes in the following areas: …

A. Knowledge and Understanding

Knowledge and Understanding of:	Teaching/Learning Methods and Strategies	Assessment
1. Object-oriented programming language concepts; other programming paradigms; syntax and semantics; top-down development; programming to satisfy designs. 2. Program design concepts, methods, and notations; object-oriented design and other design paradigms; algorithms; design patterns. 3. Databases; logical and physical database design; database query languages. 4. The concepts underpinning distributed systems and networks. 5. The concepts underpinning World-Wide Web technology and web-based application development. 6. Design and analysis of a variety of classes of algorithms. 7. The architecture and main components of computers. 8. A range of software development methods, e.g.: OOA, OOD, and OOP; and structured methods. 9. Discrete mathematics, propositional logic, and predicate logic. 10. Professional, ethical, and social values 11. The commercial context of software development 12. Formal systems: syntax, semantics, and translation between formal systems 13. Artificial intelligence concepts, notations, and Methods: including declarative languages, Deduction, and neural nets	The award is designed to introduce both the main concepts and topics of computer science, such as the design and analysis of algorithms and AI, and the knowledge and understanding necessary to engage, from the beginning, in appreciating and tackling computer-science problems. Students are introduced at each level to modules that develop a gradually increasing appreciation of the main concepts of computer science (6, 12, 13) At level-1, the context in which these issues reside is introduced, but the in-depth understanding of large, complex, problems essentially starts with level-2 study and then continues into level-3 At level 1, knowledge and understanding of topics 1, 2, 5 and 6 (object-oriented programming language concepts; program design concepts; concepts underpinning world-wide-web technology; design and analysis of algorithms) is introduced on modules which explore the general concepts, components and issues, positioning them in the computing environment. The general understanding of topics 1, 2 and 6 is built on with more in-depth knowledge and specific understanding of application at higher levels. Topic 7, “The architecture and main components of computers.” is taught only at level 1 providing, for this award, fundamental technical knowledge and understanding.	Most of the knowledge and understanding outcomes are assessed by examination (1, 2, 3, 4, 5, 6, 8, 9, 12 and 13). In addition, a variety of other assessment instruments are used to assess these outcomes, including the following: • Individual assignment project (2, 3, 4, 5, 8 and 13) • Group assignment project (5) • Extended individual project (1, 2, and 8) • Tutor appraisal (1, 2, 5 and 8) • In class test (7 and 9) • Critical review (1, 2, and 8) • Essay (10) • Presentation (10) • Portfolio of practical work (12)

Knowledge and Understanding of:	Teaching/Learning Methods and Strategies	Assessment
	At level-2 the knowledge and understanding of computing continues with an expansion into broader and larger issues, such as, the design of, and methods of building large software systems. The complexity and design of such systems is addressed in all level-2 modules. Moreover, in-depth knowledge and understanding of topics 2-4 (Program design concepts; Databases; Concepts underpinning distributed systems and networks.) is delivered in these modules. At level-2, knowledge of topic 1 is assumed but will be consolidated by constant review and usage. At level-3 students are able to obtain in-depth knowledge in a number of computing areas of their own choosing. On all modules, at all levels, the learner is encouraged to undertake independent reading both to supplement and consolidate what is being taught/learnt and to broaden their individual knowledge of the subject.	

B. Intellectual Skills

Intellectual Skills	Teaching/Learning Methods and Strategies	Assessment
1. Critical Thinking 2. Analysis 3. Synthesis of different types of information 4. Evaluation 5. Problem Solving 6. Appreciate problem contexts 7. Balance conflicting objectives 8. Construction of logical arguments 9. Discussion and debate about technical subjects with peers	At all levels students are required to bring together knowledge and skills acquired in several modules and hence determine new ways of working. As the student progresses, the need to synthesise (3) ever-greater volumes of information and approaches into a coherent approach is developed and consequently so is their critical thinking (1) as well as their ability to discuss and debate technical subjects with peers (9). At level-1 Analysis (2), Evaluation (4) and Problem Solving (5) are developed on small-scale problems in various programming activities in a number of modules. Here the focus is on understanding the problem and then solving it free from the environmental implications of real-world problems and without the need to examine alternatives and to balance conflicting goals. At level-2 there is a move away from small-scale problems to the design of larger scale systems. With this comes the need to evaluate (4) alternative methods and designs and to balance conflicting objectives (7). Level-3 sees the move to specific application examples and with it the need to appreciate problem contexts (6) is developed as well as striking the right balance when facing conflicting objectives (7). The skill of constructing a logical argument is inculcated in students in part when they develop cases to support decisions they have made to resolve conflicting objectives (8).	Programming of complex software requires demonstration of all of the intellectual skills. At level-1 the focus in programming coursework assessment, undertaken in a number of modules, is on the skills of Analysis (2), Evaluation (4) and Problem Solving (5). At level-2 and level-3 this branches out to include all the remaining skills. Many of the coursework assessments and exam papers include elements of programming work. Independent reading is used to enable students to focus on their own areas of interest and, in the process, subsequent reports, essays, and examinations allow skills 1 – 4 to be assessed. Design-work, even when not implemented in a programming language, requires demonstration of skills 1, 2, 5, 6, 7 and a number of coursework assessments and exam questions are devoted to such work. Many of the assignments and the individual project require students to express logical arguments, 8. Finally, all of the examinations assess skills 1-4 whist skills 5-7 are covered in many exams.

C. Subject, Professional and Practical Skills

Subject/Professional/Practical Skills	Teaching/Learning Methods and Strategies	Assessment
Students will be able to: 1. Write programs that conform to designs 2. Create high-level and low-level designs that correspond to stated requirements 3. Design, build, and deploy databases to meet application requirements 4. Perform adequate tests on programs 5. Elicit and express requirements for software systems 6. Build web-based systems 7. Employ a range of tools and notations to support the activities listed here: e.g. editors compliers, design workbenches, HTML, CGI, Java, etc. 8. Design algorithms using standard techniques; evaluate and compare algorithms with regard to domain problems; use mathematical techniques to analyse algorithm complexity; apply algorithms appropriately to real-world tasks.	Throughout the program, the skills listed are developed through a combination of the following devices: • Theoretical discussion • Practical laboratory-based work • Classroom-based tutorial exercises • Directed self-study Many of the skills listed are introduced at level-1 and then drawn into sharper focus at level-2, and deepened at level-3. The general teaching/learning method is to impart these practical/professional skills by a process of moving from an overview of what is required to a specific application of an individual skill at a higher level.	The two main forms of assessment of the subject, professional and practical skills are as follows: • Extended individual project (1, 2, 4, 5, 7) • Practical component of individual assignment projects (1 – 9) In addition, other assessment instruments are used to assess some of the skills: • Examination (1, 2, 3, 5, 8, 9) • Portfolio of practical work (9) • Group assignment project (5)

D. Transferable Skills and Other Attributes

Transferable Skills and Other Attributes	Teaching/Learning Methods and Strategies	Assessment
1. Communication skills: to communicate orally or in writing.	1. Communication skills are developed through a variety of methods and strategies including the following: ♦ Students maintain laboratory log books ♦ Students participate in electronic conferences, workshops, and groupwork sessions. ♦ Students participate in discussion tutorials ♦ Students present research topic findings in tutorials Students participate in individual tutorials	1. Communication skills are assessed mainly by examination, but also by in-class tests, essays, presentations and poster presentations. 2. The other skills are assessed through a number of similar
2. Self-management skills: to manage one's own time; to meet deadlines; to work with others.	2. Self-management skills are developed through a variety of methods and strategies including the following: ♦ Students conduct self-managed practical work ♦ Students participate in practically-oriented tutorial laboratory sessions ♦ Students work through practical work-sheets in teams ♦ Students practice design and programming ♦ Students participate in electronic conferencing tutorials Students participate in electronic groupworking tutorials	instruments including the following: ♦ Individual and group projects ♦ Practical assignments ♦ Portfolio of exercises 3. In addition self-management skills are assessed by both peers and tutors.
3. IT skills in context: to use software tools in the context of application development.	3. IT skills in context are developed through a variety of methods and strategies including the following: ♦ Students conduct self-managed practical work ♦ Students participate in experimental investigation tutorials ♦ Students work through practical work-sheets in teams ♦ Students make use of online teaching materials ♦ Students use a range of system development tool, methods, and packages ♦ Students are encouraged to practice programming to extend their skills ♦ Students make sustained use of the internet	
4. Logical reasoning skills: To undertake analysis and interpretation of information in the context of the Computing discipline.	4. Logical reasoning skills are developed through a variety of methods and strategies including the following: ♦ Students develop problem-solving programs ♦ Case-Studies are used to explore design issues with students ♦ Students practice design and programming Students sketch designs of larger systems	

5. Problem formulation: To express problems in appropriate notations.	5. Problem formulation skills are developed through a variety of methods and strategies including the following: ♦ Students develop problem solving programs ♦ Students practice design and programming Students sketch designs of larger systems	
6. Progression to independent learning: To gain experience of, and to develop skills in, learning independently of structured class work. For example, to develop the ability to use on-line facilities to further self-study.	6. Progression to independent learning is developed through a variety of methods and strategies including the following: ♦ Students are encouraged to practice programming to extend their skills ♦ Students develop problem-solving programs ♦ Students are encouraged to research relevant topics ♦ Students are encouraged to use the library, the internet and other online facilities to discover information and broaden knowledge ♦ Students are encouraged to articulate and reflect upon their own ideas and experiences Students negotiate the content and structure of their individual projects with tutors	
7. Comprehension of professional literature: to read and to use literature sources appropriate to the discipline to support learning activities.	7. Comprehension of professional literature is developed through a variety of methods and strategies including the following: ♦ Students are encouraged to access online material ♦ Material is recommended to the students in module syllabi and by tutors Students are required to research and refer to appropriate literature in assignments and the individual project	
8. Information access: to understand basic techniques for structuring and thereby accessing information.	8. Skill eight is developed through a variety of methods and strategies including the following: Students develop a database system in laboratory sessions	

Section 4: Programme Structure

Note: This structure is indicative and subject to change

BSc (Hons) Computer Science

Year 1					
UFCE46-20-1 Introduction to Program Development	UFCE47-20-1 Systems Development	UFEEHV-20-1 Computer Systems	UFCE48-20-1 Computer Science Concepts	UFQEFY-20-1 Analytical Modelling	UFCE3H-20-1 Computational Intelligence

Year 2					
UFCE4A-20-2 Data Structures and Databases	UFEEHX-20-2 Computer Networks & O/S	UFCE4B-20-2 Software Design	Option: 1 20 CREDITS	Option: 2 20 CREDITS	Option: 3 20 CREDITS

Year 2P
Option: Placement 0 - 120 CREDITS

Year 3			
UFCE3B-40-3 Computing Project	UFCE3J-20-3 Agent-Based Computing	UFIE9K-10-3 Professional, Legal and Commercial Issues	Option: 4 50 CREDITS

Option: 1 choose from:	
UFCE4C-20-2	Declarative Programming
UFCE4D-20-2	Symbolic Processing
UFCE4E-20-2	Subsymbolic Processing

Option: 2 choose from:	
UFCE4F-20-2	Graphics Programming
UFCEKP-20-2	Client-Server Programming

Option: 3 choose from:	
UFCE49-20-2	Software Engineering
Choose modules from	Option: 1
Choose modules from	Option: 2

Option: Placement **Students who are taking a placement year can also take level 3 optional module UFPES8-20-3 Professional Practice whilst on placement. Students choosing this option will only need to accumulate 100 credits for their final year of study.** **choose from:**	
UFPEJH-120-P	Industrial Placement

Option: 4 **Please note - UFPES8-20-3 is ONLY available to students who are taking a placement.** **choose from:**	
UFCE49-20-2	Software Engineering
UFCE4P-10-3	Object Oriented Databases
UFCE4Q-10-3	Distributed and Parallel Databases
UFCE4R-10-3	Intelligent Systems
UFCE4T-10-3	Interface Engineering
UFCE4X-10-3	Software Technologies for the Web
UFCE4Y-20-3	Component Based Development
UFEEJ6-10-3	Advanced Distributed Systems
UFCE3K-20-3	Machine Learning
UFEEJB-20-3	Advanced Systems Administration
UFCEMU-20-3	Cryptography and Coding Systems
UFPES8-20-3	Professional Practice
Choose modules from	Option: 1
Choose modules from	Option: 2

Please Note – Students are responsible for ensuring that their completed programme will contain the required number of credits at the appropriate levels.

Generated from CEMS FOLD on 12th November 2007

PLEASE NOTE: REFER TO THE FACULTY ON-LINE INFORMATION SYSTEM FOR UP-TO-DATE STRUCTURE INFORMATION
http://www.cems.uwe.ac.uk/exist/index.xql

Section 5: Entry Requirements

The university's minimum requirements for entry to a degree will apply. In addition entrants will be required to have:

- Mathematics at GCSE Grade C or equivalent.
- Mathematics, Computing or a Science at A2 level
- Prior knowledge of a programming language.

Section 6: Assessment Regulations

a) **MAR**

Section 7: Student Learning: Distinctive Features and Support

Within the Faculty of Computing Engineering and Mathematical Sciences, student learning will be supported in the following distinctive ways:

- through provision of a large Open Access Laboratory (3P10) containing 50 machines that provide students with access to a wide range of computer-based applications;
- through provision of nine other, frequently available, computer laboratories that provide similar access;
- through provision of the CEMS System Support Helpdesk that provides a range of support for learning to students including:
- support for a wide range of applications used by the students;
- help in the form of Assistants who are trained to resolve many common student problems;
- and help in the form of a large set of “Helpsheet Documents”, developed over a number of years, that cover a variety of common student requests for information;
- in level-3 modules there is scope for engagement with current leading-edge research undertaken by researchers within the University.

Section 8 Reference Points/Benchmarks

- *Subject benchmarks (QAA Unit …)*
- *University teaching and learning policies:*
- *staff research projects:*
- *employer interaction/feedback:*

The QAA Subject Benchmark Statement for Computing was published in 2000, and is applicable to this proposal. The design team has considered it in drawing up the structure of the programme, and is of the view that the proposal falls clearly within the scope of the benchmarks, as regards curriculum, teaching and learning, and the benchmarking standards themselves.

The benchmarks (paragraph 2.1) identify a range of types of degrees in computing. At one extreme is a programme that "*covers a wide range of topics spanning the entire area of computing".* At the other programmes that "*take one very specific aspect of computing and covers it in great depth"*. This programme resides in the middle of these two extremes providing relatively detailed coverage of a moderately broad subset of computing topics and embraces the three key ideas:

- Development of computing systems;
- Importance of specialism and position within a broader context;
- Balance between theory and practice.

The benchmarks establish a set of Principles of Course Design (paragraph 3.1). This programme, whilst first developed prior to the writing of the benchmarks, nevertheless satisfies these design principles and continues to be revised bearing them in mind.

The benchmarks also contain (section 5) statements of the standards expected of graduates at both modal and threshold levels. The team is of the view that graduates of the programme will be able to meet the required standards, and indeed have done so on earlier versions of the programme.

This specification provides a concise summary of the main features of the programme and the learning outcomes that a typical student might reasonably be expected to achieve and demonstrate if he/she takes full advantage of the learning opportunities that are provided. More detailed information on the learning outcomes, content and teaching, learning and assessment methods of individual modules are to be found in the module specifications.