Half Award in Computing

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) Computing &
Default award title	BSc Computing &
Interim award title	Diploma of Higher Education, Certificate of Higher Education
Modular Scheme title (if different)	MAR
UCAS code (or other coding system if relevant)	
Relevant QAA subject benchmarking group(s)	Computing
On-going/valid until* (*delete as appropriate/insert end date)	
Valid from (insert date if appropriate)	

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 half-award in Computing has the following general aims:

1. To prepare students for computing careers in business, industry, and commerce, or in organisations with a significant in-house IT management culture.

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 progressing to study for higher degrees in computing.

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

The half-award in computing has the following specific aims:

1. To provide an approach to teaching and learning which maximises the deployment of e-based facilities, like electronic discussion boards, within award modules in order to improve student access to materials and thus provide support for learning.

2. To develop the students' ability to make an immediate contribution to companies engaged in electronic commerce and/or web-based development.

3. To develop the students' understanding of the importance of project planning in any domain, though with particular reference to the development of software projects.

4. To encourage the discerning use of reference material from a variety of sources.

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. Object-oriented and related 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. Electronic commerce; architectures and components of commercial applications based upon www technology; technical and management issues. 7. The concepts underlying the reuse of components and framework in software development; related research issues. 8. The architecture and main components of computers. 9. The concepts underpinning user interfaces; good design practice; notation issues; user interface evaluation.	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. The programme of study is designed as to introduce the knowledge and understanding necessary to engage, from the beginning, in appreciating and solving small-scale problems. At level 1, the context in which these issues reside is introduced but the in-depth understanding of large, complex, real-world problems essentially starts with level 2 study. At level 3, we continue to increase our in-depth knowledge and understanding of in-depth to technical solutions of real-world problems for topics pertinent to the present state of the industry. At level 1, knowledge and understanding of topics 1-6 (Object-oriented programming language concepts; Program design concepts; Object-oriented and related databases; Concepts underpinning distributed systems and networks; Concepts underpinning World-Wide-Web technology; Electronic commerce) is introduced on two modules which explore the general concepts, components and issues, positioning them in the computing environment. The general understanding of topics 1-6 is built on with more in-depth knowledge and specific understanding of application in further levels. Topic 8, “The architecture and main components of computers.” is taught only at level 1 providing, for this award, the delimiters of a sufficient technical knowledge and understanding. Topics 7 & 9 (“Concepts underlying the reuse of components and framework in software development”, and “Concepts underpinning user interfaces; good design practice; notation issues; user interface evaluation.”) have only a cursory mention at level 1 although the more astute learner will find consistent references to relevant knowledge.	Testing of the knowledge base is through: Assessed coursework (topics: 1, 2, 3, 4, 6, 8, 9); Assessed practical work (topics: 1, 2); Examination (topics: 1, 2, 3, 4, 5, 7, 6, 8, 9); Peer and tutor evaluation (topics: 2, 5); Group coursework/project (topics: 2, 5, 6); Portfolio of exercises (topics: 1, 4, 7); Poster presentation (topics: 1, 4, 7); Internet/online assessment (topics: 1, 5).

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 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; Object-oriented and related 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. The development of specialised and more specific knowledge and understanding emerges level 3 where half-modules, seen for the first time, allow in-depth focus on advanced topics. In particular, topics 3 (Object-oriented and related databases), 5 (Concepts underpinning World-Wide Web technology and web-based application development.), 6 (Electronic commerce) and 9 (The concepts underpinning user interfaces; good design practice; notation issues; user interface evaluation.) each have a module dedicated to delivering the necessary technical knowledge and understanding. The remaining module primarily delivers the knowledge and understanding topic 7 (concepts underlying the reuse of components and framework in software development), and further consolidates topics 1 and 4.	

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	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). 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).	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 levels 2 and 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 asses skills 1-4 in the submitted reports, essays and exam answers. 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. 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 databases to meet application requirements 4. Create user interfaces for a variety of applications 5. Perform adequate tests on programs 6. Know how to use existing components and frameworks to build new applications 7. Build web-based systems 8. Employ a range of tools and notations to support the activities listed above: e.g. editors, compilers, design workbenches, HTML, CGI, Java etc..	Throughout the program, the skills listed are developed through a combination of theoretical discussion, practical laboratory based work, classroom based tutorial exercises and directed self-study. Many of the skills listed (1,2,3,5,6,8) are introduced at level 1 and then drawn into sharper focus at levels 2 and 3. The general teaching/learning method is therefore 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. Some very specific skills (4, 7) are introduced at level 3. These are underpinned by the more generalised capabilities (1, 8) that are practised throughout the levels in most of the modules that contribute to the award.	The possession of these skills is demonstrated both by the development of a practical piece of coursework (software) and by examination. The practical nature of the skills to be acquired means that some are specifically addressed by particular modules (3, 4, 6, 7). The more generic skills (1,2,5,8) are assessed across the modules. For example, the module ‘Interface Engineering’ requires the students to develop a particular user interface (4) as part of the assessment whilst the examination allows students to demonstrate that they have grasped the underlying concepts that inform the professional development of such an artefact. Skills such as conformance to design and requirements (1, 2) and the construction of adequate testing strategies (5) are fundamental to professional software development of any sort and thus contribute to the assessment of all the practical work produced.

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, including, for instance, the results of technical investigations, to peers and/or to "problem owners".	1. Skill one is 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. Skill one is demonstrated mainly by examination, but also by poster presentation. 2. Skills two through eight are demonstrated by a number of similar instruments including the following: ♦ Individual and group projects ♦ Practical assignments ♦ Portfolio of exercises 3. In addition skill two is assessed by both peers and tutors
2. Self-management skills: to manage one's own time; to meet deadlines; to work with others having gained insights into the problems of team-based systems development.	2. Skill two is 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	
3. IT skills in context: to use software tools in the context of application development.	3. Skill three is 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 are encouraged to practice programming to extend their skills	

Transferable Skills and Other Attributes	***Teaching/Learning Methods and Strategies***	***Assessment***
4. Logical reasoning skills: To undertake analysis and interpretation of information in the context of the Computing discipline.	4. Skill four is 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. Skill five is 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. Skill six 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 online facilities to discover information	
7. Comprehension of professional literature: to read and to use literature sources appropriate to the discipline to support learning activities.	7. Skill seven is developed through a variety of methods and strategies including the following: ♦ Students are encouraged to access online material	
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 for Half Award in Computing
Note: This structure is indicative and subject to change

Object Oriented Databases UFCE4P-10-3	Interface Engineering UFCE4T-10-3	Software Technologies for the Web UFCE4X-10-3	Component-Based Development UFCE4Y-20- 3	E Commerce Special Interest Groups UFIE86-10-3

Year 2 P (Industrial Placement Year)

Data Structures and Databases UFCE4A-20-2	Software Design UFCE4B-20-2	Distributed Systems & Networks UFIE8X-20-2
20 credits Option 1	Systems Development UFCE47-20-1	20 credits Option 2

Option 1 taken from	
****6	Students not taking AI half award MUST choose UFCE46-20-1
****9	Students taking AI half award MUST choose UFCE4V-20-1
UFCE46-20-1	Introduction to Program Development
UFCE4V-20-1	Programming for Mathematics

Option 3 taken from	
****8	Students not taking Internet Systems half award MUST take UFIE8W-20-1
UFCE48-20-1	Computer Science Concepts
UFIE8Q-20-1	Information Systems Application Context
UFIE8W-20-1	Information Technology

Section 5: Entry Requirements

Students must achieve at least 200 points in the tariff point range. This equates to 3 Cs at A-level. Equivalent qualifications, like Baccalaureate or Irish Higher, will be acceptable. Students must have a pass in GCSE Mathematics at a minimum of Level C.

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.

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 them in drawing up the structure of the proposed half-degree, 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 (para 2.1) identify a range of types of degrees in computing, from (at one extreme) a programme which "*covers a wide range of topics spanning the entire area of computing"* to (at another extreme) programmes which "*take one very specific aspect of computing and covers it in great depth"*. This proposal is closer to the first of these extremes. Nevertheless it does allow students to recognise the importance of speciality areas, through the choice of Level 3 modules.

The benchmarks recognise (para 3.3) that diversity of provision is to be encouraged, and hence that joint degrees have an important place. Nevertheless there are inevitably constraints on the breadth of coverage of the subject possible within a "half-degree". The design team has faced these constraints in the context of the principles of course design as set out in the benchmarks (para 3.1), and it believes that it has successfully met them all to the extent that it is possible to do so within the half-degree structure.

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 proposed programme will be able to meet the required standards, albeit in some cases to a lesser depth than would be expected of a graduate in a full honours degree in computing.