Half-Award in Artificial Intelligence

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) Artificial Intelligence &
Default award title	
Interim award title	BSc 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 Artificial Intelligence has the following general aims:

1. To prepare students for careers in organisations with an innovative research and development culture especially in-house development of novel software.

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 Computer Science and in particular Artificial Intelligence.

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

The half-award in Artificial Intelligence has the following specific aims:

1. To provide insight into, and practical skills in, the creation of intelligent computer systems. This involves understanding the nature of intelligence in both computers and people, and using the resulting insight to build smart technologies.

2. To develop the students' ability to make an immediate contribution to companies engaged in the design of intelligence enabling technologies of various kinds.

3. 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.

4. To encourage the discerning use of AI 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 concepts; syntax and semantics; programming to satisfy designs. OO Programming Language.
2. Program design concepts, methods, and notations; object-oriented design paradigms; algorithms; design patterns.
3. Simple models of computation
4. Be aware of the possibility of proving the correctness of algorithms.
5. The biological basis for neural networks, fuzzy systems, evolutionary computation and swarm intelligence.
6. Understand the essential concepts of the theory of Evolutionary Computation and the inner workings of genetic algorithms.
7. The principles behind functional and logic programming and their advantages over imperative languages, and their problems and limitations.
8. Appreciate that building intelligent systems and modeling human cognition are interrelated but different disciplines that have separate objectives and that a number of different paradigms and architectures exist for intelligent systems.
9. Be aware of some of the more important and useful neural network architectures and appreciate the applications of neural computing in industry.
	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 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 by contrasting principles of algorithmic computation with those of intelligent systems. The in-depth understanding of large, complex, real-world problems essentially starts with level 2 study. Level 3 continues to increase the in-depth knowledge and understanding of technical solutions to real-world problems.
	At level 1, knowledge and understanding of topics 1-6 (Object-oriented programming language concepts; models of computability, and evolutionary computing) are introduced on three modules that explore the general concepts and issues, positioning them within a computer science framework. More in-depth knowledge and specific understanding of all topics follows in subsequent levels.
		Testing of the knowledge base is through:
		Assessed coursework (topics: 3, 4);
		Assessed practical work (topics: 1, 2, 6, 7, 9, 10, 11, 12);
		Examination (topics: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13);

Knowledge and Understanding of:	Teaching/Learning Methods and Strategies	Assessment
10. Understand the fundamental theories of neural networks, fuzzy logic systems and associated learning algorithms. 11. Explain how agents differ from other software technologies (e.g. object-orientation), and describe how multi-agent communication and cooperation can effectively solve certain application domain problems. 12. Identify the range of machine learning techniques and how these relate to traditional approaches to artificial intelligence explaining their applicability to particular problems. 13. Know the location and nature of a variety of resources invaluable to the practitioner and explore in depth a significant emerging topic of interest to the AI community.	At level 2 the deepening of the knowledge and understanding of AI continues with an expansion into broader and larger issues, such as, alternative programming paradigms, modeling human cognition and applications of neural networks. The development of specialized and more specific knowledge and understanding continues at level 3. Use of some half-modules for the first time allows in-depth study of sharply focused advanced topics such as Fuzzy-logic and a choice of topics within the Intelligent Systems module. Use of some standard sized modules provides fuller coverage of major topics such as Machine Learning and Agent-based Systems.	

B. Subject Specific Skills

Subject Specific 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. Employ a range of tools and notations to support the software development activities e.g. editors, compilers, support tools and software development environments. 4. Express the syntax of formal languages in terms of productions. 5. Describe algorithmic behaviour in appropriate formal terms. 6. Perform worst-case analysis on simple algorithms and describe performance using Big-O notation.	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. A number of the skills listed (1, 2, 3, 6, 7) 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 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 (9, 12, 13, 14) are introduced at level 3. These are underpinned by the more generalised capabilities (1, 2, 3, 7,) 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 many of the skills to be acquired means that particular modules specifically address skills 4, 6, 7, 8, 9, 10, 11, 12, 13, and 14. The more generic skills 1, 2, 3, and 5 are assessed across the modules.

Subject Specific Skills	Teaching/Learning Methods and Strategies	Assessment
7. Develop, train and test a neural network and use an evolutionary algorithm to solve an appropriate problem. 8. Write both functional and logic programs in appropriate languages. 9. Use a logic language to augment an unsophisticated translation program. 10. Develop and test simple architectures using graphical neural network development tools. 11. Access and evaluate articles drawn from the latest publications concerned with various themes and research strands in intelligent systems. 12. Use a simulation environment for Neuro-fuzzy systems development. 13. Design and implement intelligent autonomous agents in an OO programming language such as Java. 14. Apply suitable machine learning techniques to new problems using appropriate software to develop the solution.		For example, the module Agent-based Computing requires the students to design and implement intelligent autonomous agents in an OO programming language such as Java (13) as part of the coursework assessment. The examination allows students to demonstrate that they have grasped the underlying concepts that inform the development of such an artifact. Skills such as conformance to requirements and design and (1, 2) are fundamental to professional software development of any sort and thus contribute to the assessment of all the practical work produced.

C. Cognitive (Intellectual) Skills

Cognitive (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 relatively trivial problems to the consideration of larger scale more complex 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 yet more sophisticated techniques and more complex 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, 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 whilst skills 5-7 are covered in many exams.

D. Key (Transferable) Skills

Key (Transferable) Skills	Teaching/Learning Methods	Assessment
1. Communication skills: to communicate orally or in writing.	1. Skill one is developed through a variety of methods and strategies including the following: ♦ 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	All of the skills are demonstrated in varying degrees in all of the employed assessments with the exception of teamwork, which is covered in some of the coursework. It would be impossible to progress to completion on the half-award without demonstrating a basic competence in all of these skills.
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 software 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 and electronic group-working	
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. Problem Formulation and Decision-Making: To undertake analysis and interpretation of information and express problems in appropriate notations.	4. Skill four is developed through a variety of methods and strategies including the following: ♦ Students develop problem solving programs ♦ Students practice design and programming in a number of different languages ♦ Students sketch designs of larger systems	
5. 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.	5. Skill five is developed through a variety of methods and strategies including the following: ♦ Students are encouraged to practice programming to extend their skills ♦ Students are encouraged to research relevant topics ♦ Students are encouraged to use online facilities to discover information	
6. Awareness of professional literature: to read and to use literature sources appropriate to the discipline to support learning activities	6. Skill six is developed through a variety of methods and strategies including the following: ♦ Students are encouraged to access online material ♦ Students review the literature for discussion in tutorial classes and electronic conferences	
7. Teamwork: to be able to work as a member of a team; to be aware of the benefits and problems which teamwork can bring	7. Skill seven is developed through a variety of methods and strategies including the following: ♦ Students are required to participate in electronic conferences ♦ Students will develop software in small groups	

Section 4: Programme Structure

Note: This structure is indicative and subject to change

Structure for the Artificial Intelligence Half-Award [Updated on 10/12/02]

Agent-based Computing UFCE3J-20-3 (UFC007S3)	Machine Learning UFCE3K-20-3 (UFC008S3)	Intelligent Systems UFCE4R-10-3 (UQC137H3)	Neural Networks & Fuzzy Systems UFEE7Q-10-3 (UFE023H3)	Year 3
Year 2P (Industrial Placement)				
Declarative Programming UFCE4C-20-2 (UQC114S2)	Symbolic Processing UFCE4D-20-2 (UQC115S2)	Sub-symbolic Processing UFCE4E-20-2 (UQC116S2)		Year 2
Program Development UFCE45-20-1 (UQC101S1)	Computer Science Concepts UFCE48-20-1 (UQC106S1)	Computational Intelligence UFCE3H-20-1 (UFC006S1)		Year 1

□ Core modules

Section 5: Entry Requirements

The university's minimum requirements for entry to a degree will apply, and in addition entrants will be required to have evidence of achievement in Mathematics at GCSE Grade C or equivalent.

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 both researchers within the University and at collaborating institutions.

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 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 (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 proposal is closer to the second of these extremes.

The benchmarks recognise (paragraph 3.3) that diversity of provision is to be encouraged, and hence 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 as part of the course design as set out in the benchmarks (paragraph 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 Artificial Intelligence.

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.

We have undertaken numerous projects in applications of AI with collaborating organisations at companies such as HP, Motorola, BT, LloydsTSB, Siemens and others. This has made us aware of the growing requirement for skills in these technologies.