Module Specification

Object Oriented Software Design and Development II

Version: 2021-22, v1.0, 12 Jul 2021

Contents

Module Specification **1**

Part 1: Information **2**

Part 2: Description **2**

Part 3: Teaching and learning methods **4**

Part 4: Assessment **5**

Part 5: Contributes towards **7**

Part 1: Information

Module title: Object Oriented Software Design and Development II

Module code: UFCFYM-15-2

Level: Level 5

For implementation from: 2021-22

UWE credit rating: 15

ECTS credit rating: 7.5

Faculty: Faculty of Environment & Technology

Department: FET Dept of Computer Sci & Creative Tech

Partner institutions: None

Delivery locations: University Centre Weston

Field: Computer Science and Creative Technologies

Module type: Standard

Pre-requisites: None

Excluded combinations: None

Co-requisites: None

Continuing professional development: No

Professional, statutory or regulatory body requirements: None

Part 2: Description

Overview: Students will learn the basic concepts of software design, data structures, programming, problem solving, programming logic, and fundamental software design techniques. This will include a review of traditional and contemporary software development methods including agile development. They will develop a holistic view of software engineering, practice including gathering requirements, designing a

solution, implementing a solution in a programming language, testing the completed application and deploying the solution to end users

Features: Not applicable

Educational aims: The purpose of this topic is to introduce students to the fundamental concepts of systems development through programming, computational thinking and data structures. They will analyse models of application development so that they can understand the key processes related to building functioning applications and appreciate the complexity of application development.

Outline syllabus: The syllabus includes:

Demonstrate an understanding of object-oriented concepts

Outline the general trends in software development, and identify the perceived advantages of object-oriented techniques (e.g. modularity, encapsulation, reuse, iterative development, interactivity, greater client involvement in design. Identification of objects, classification, inheritance, polymorphism)

Perform object-oriented analysis and design

Develop modelling techniques appropriate to object-oriented design (e.g. object diagrams, class diagrams, use cases, state diagrams, scenarios, sequence diagrams, collaboration diagrams, CRC cards and appropriate use of data dictionary)

Emphasis will be placed on UML and the use of a Case Tool

Identify and select the most appropriate paradigm for a business case study, explaining why that paradigm will be the most relevant (e.g. Event Driven, Procedural, Object-Oriented)

Create a comprehensive test plan that utilises unit tests, ensuring errors are analysed to help correct errors

Develop complex software using two cohesive languages while utilising at least two

different paradigms

Design appropriate usability study

Adhere to industry standards (e.g. Code Documentation, Naming Conventions)

Object-oriented Program Development Code features

Style and structure

Syntax and semantics

Prepare code for classes to be re-used in other applications

Object-oriented Program Evaluation Conduct tests

Part 3: Teaching and learning methods

Teaching and learning methods: Introductory lectures are supported by seminars, case studies, visits and practical workshops. In addition, this module will be supported by interactive forums and learning tools.

Independent learning includes hours engaged with essential reading, case study preparation, assignment preparation and completion. Study time will be organised each week with a series of both essential and further readings and preparation for practical workshops.

This unit practically based and designed to ensure that students understand and develop their skills in advanced programming techniques. Students will use the object-oriented facilities within C++ as a vehicle for this.

Module Learning outcomes:

MO1 Apply procedural and/or object-oriented programming techniques.

MO2 Identify, explain, and use appropriate business and technical requirements to select suitable solutions.

MO3 Create data models and software designs to effectively communicate understanding of the programme.

MO4 Implement, test, and debug complex software solutions to meet a requirements specification.

MO5 Develop complex software solutions and software modifications to specified requirements.

MO6 Test code and analyse results to correct errors found using unit testing.

MO7 Apply underlying concepts and the principles of best practices and standards.

Hours to be allocated: 150

Contact hours:

Independent study/self-guided study = 114 hours

Face-to-face learning = 36 hours

Total = 150

Reading list: The reading list for this module can be accessed at readinglists.uwe.ac.uk via the following link https://rl.talis.com/3/uwe/lists/191A5CA5-84FA-5EF6-E16C-CCD8C3E9D807.html

Part 4: Assessment

Assessment strategy: This module is assessed by a combination of techniques: an examination (1.5 hours) and a practical build.

Component A – Exam

Students will be required to sit a 90 minute exam that will require knowledge of the

both object-oriented and procedural programming techniques.

Students will be required to perform object-oriented design using a taught methodology, against a business specification. They will need to employ data models and designs they have been taught such as sequence and state diagrams.

Students will be required to create a structurally sound program using their own selected paradigm (event driven, procedural or object-orientated) to a set of business requirements.

Component B – Practical Build
Students will be given a business specification from which they will produce a solution. They will need to design their systems and apply their knowledge of the development lifecycle models to create a sound system.

The task will include development, implementing, testing and debugging. The testing of the program will need to be robust and thorough, using unit testing. Students will be required to track the errors found by unit testing, and evaluate, applying corrections as required. The program will need to be fully documented and conform to industry standards.

Opportunities for formative assessment exist for the assessment strategy used. Verbal feedback and written feedback is given to all students providing a personal platform for improvement.

Assessment components:

Examination - Component A (First Sit)
Description: Examination (1.5 hours)
Weighting: 50 %
Final assessment: Yes
Group work: No
Learning outcomes tested: MO2, MO3

Portfolio - Component B (First Sit)

Description: Design, implement, test and correct a problem specification.

Weighting: 50 %

Final assessment: No

Group work: No

Learning outcomes tested: MO1, MO4, MO5, MO6, MO7

Examination - Component A (Resit)

Description: Examination (1.5 Hours)

Weighting: 50 %

Final assessment: Yes

Group work: No

Learning outcomes tested: MO2, MO3

Portfolio - Component B (Resit)

Description: Design, implement, test and correct a problem specification.

Weighting: 50 %

Final assessment: No

Group work: No

Learning outcomes tested: MO1, MO4, MO5, MO6, MO7

Part 5: Contributes towards

This module contributes towards the following programmes of study:

Digital and Technology Solutions (Software Engineer) {Apprenticeship-UCW} [Sep][FT][UCW][4yrs] BSc (Hons) 2020-21