



## MODULE SPECIFICATION

Part 1: Information			
Module Title	Fundamentals of Software Development		
Module Code	UFCFQM-30-1	Level	Level 4
For implementation from	2019-20		
UWE Credit Rating	30	ECTS Credit Rating	15
Faculty	Faculty of Environment & Technology	Field	Computer Science and Creative Technologies
Department	FET Dept of Computer Sci & Creative Tech		
Module type:	Project		
Pre-requisites	None		
Excluded Combinations	None		
Co- requisites	None		
Module Entry requirements	None		

Part 2: Description
<p><b>Overview:</b> The purpose of this topic is to introduce the apprentices 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.</p> <p><b>Educational Aims:</b> Apprentices 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.</p> <p><b>Outline Syllabus:</b> Development lifecycles and their differences (e.g. Waterfall model, Agile)</p> <p>Application of programming principles to business requirements (e.g. design, code, test, correct, deploy and document) including the approach required</p> <p>Role of legacy systems in program development</p>

## STUDENT AND ACADEMIC SERVICES

Programming paradigms and the reasoning behind selecting it for business requirements (e.g. Event Driven, Object Orientation, Procedural)

Analyse business' software requirements to create a sound solution

Develop a program using an industry recognised language and development lifecycle model (e.g. Design, implement, test and debug)

Create modifications to a created to program to meet program requirements

Testing facilities and tracking to be able to debug created program code to understand and rectify problems within the code (e.g. white box, black box, unit testing)

Usage of industry standards to create robust and efficient code and understanding the necessity of this practice for both solo and group projects

Creation of end user training (e.g. User guide, Tooltips, help facilities)

**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.

300 hours study time of which 72 hours will represent scheduled learning.

Independent learning includes hours engaged with essential reading, assignment and completion. Apprentice study time will be organised each week with a series of both essential and further readings.

This module will be based on ensuring that apprentice's practical skills are developed in programming. Every session will incorporate designated practical work to complete in order to ensure that apprentices understand and implement principles of good practice.

### Part 3: Assessment

This module is assessed by a combination of techniques: a report (3,000 words) and a practical build.

#### Component A1 – Report

Apprentices will be required to write a 3,000-word report that will identify the core concepts that a software developer will need to follow in the development of software applications. Apprentices will need to discuss the stages of a different development lifecycles and their stages.

Apprentices will be required to understand the application of programming principles throughout the stages of the lifecycle model and how to approach each stage. Within the development stage, they should also be able to identify the reasoning behind selecting a paradigm for the development task.

Apprentices will need to be able to demonstrate their understanding of why legacy systems are used within a business environment to test new or existing projects.

#### Component A2 – Practical Build

Apprentices 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 practical software design, development, implementing, testing and debugging. The testing of the program will need to be robust and thorough, using techniques such as white and black box testing and unit testing. After the testing, apprentices will be required to modify their existing code base and apply new fixes or modules from their requirements and testing. The program will need to be fully documented and conform to industry standards.

## STUDENT AND ACADEMIC SERVICES

Apprentices will be required to create end user training that should be able to be delivered with a successful system.

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

First Sit Components	Final Assessment	Element weighting	Description
Report - Component A	✓	40 %	Report (3,000 words)
Project - Component A		60 %	Fully Documented (e.g. technical, user guide, algorithms) and Implemented System
Resit Components	Final Assessment	Element weighting	Description
Report - Component A	✓	40 %	Report (3,000 words)
Project - Component A		60 %	Fully Documented (e.g. technical, user guide, algorithms) and Implemented System

### Part 4: Teaching and Learning Methods

Learning Outcomes	On successful completion of this module students will achieve the following learning outcomes:	
	<b>Module Learning Outcomes</b>	<b>Reference</b>
	Understand basic programming concepts.	MO1
	Understand programming principles including design, code, test, correct, deploy and document from supplied specifications, using agreed standards and tools.	MO2
	Understand the stages of a software development lifecycle.	MO3
	Understand the similarities and differences between agile and waterfall software development methodologies.	MO4
	Be aware of the role and position of legacy systems in organisations and how new development environments interface and integrate with them.	MO5
	Understand how teams work effectively to produce software.	MO6
	Understand software design approaches and patterns and can interpret and implement a given design.	MO7
	Analyse business and technical requirements and select appropriate solutions.	MO8
	Design, implement, test, and debug software to meet a requirement's specification.	MO9
	Select the relevant paradigm for a given set of business requirements.	MO10
	Develop moderately complex software solutions and software modifications to specified requirements.	MO11
	Test code and analyse results to correct errors found using unit testing.	MO12
	Debug own code and understand structure of programmes in order to identify and resolve issues.	MO13
	Identify and apply best practices and standards.	MO14
	Identify and implement plans for end user training and built in tool tips and help facilities.	MO15
Contact Hours	<b>Independent Study Hours:</b>	
	Independent study/self-guided study	228

## STUDENT AND ACADEMIC SERVICES

	<b>Total Independent Study Hours:</b>	228
	<b>Scheduled Learning and Teaching Hours:</b>	
	Face-to-face learning	72
	<b>Total Scheduled Learning and Teaching Hours:</b>	72
	<b>Hours to be allocated</b>	300
	<b>Allocated Hours</b>	300
Reading List	<i>The reading list for this module can be accessed via the following link:</i> <a href="https://uwe.rl.talis.com/index.html">https://uwe.rl.talis.com/index.html</a>	

<b>Part 5: Contributes Towards</b>	
This module contributes towards the following programmes of study:	