# MODULE CODE XPX1041

# MODULE TITLE Software Design

# SCHOOL/s INVOLVED IN DELIVERY None

# NAME OF COURSE(S) Certificate of Higher Education In

# Engineering

# MODULE LEADER Taha Al Jody

# LOCATION Huddersfield ISC

# MODULE TYPE Core

# CREDIT RATING 20

# LEVEL F (RQF Level 4)

# LEARNING METHODS Lecture, Practical Classes and

# Demonstrations – 42 Hours

Guided Independent study – 158 hours

# PRE-REQUISITE/S None

# RECOMMENDED PRIOR STUDY None

# CO-REQUISITE/S None

# Shared teaching None

# PROFESSIONAL BODY REQUIREMENTS None

# GRADED OR NON-GRADED Graded

# BARRED COMBINATIONS None

# SYNOPSIS

This module will introduce the concepts of abstraction techniques, modelling notations, introduction to structured and object-oriented programming, algorithm design, data structure design, and program implementation, testing, and documentation.

# LEARNING STRATEGY

Contact time typically consists of lectures and practical labs but is context specific. Lectures will introduce the theoretical foundations of modelling and object-oriented programming concepts which will include example applications. There will also be some discussion of the historical origins and current exploitation of techniques so that learners may appreciate the significance of the material to which they are being introduced.

Sessions will be used to explore and support the application of the theoretical concepts. Hands-on application to simple scenarios will be used, and learners will be encouraged to develop their ability to abstract useful information and to recognise where assumptions may have to be made.

Huddersfield International Study centre encourages students to be academic citizens who are self- directed and motivated learners. Key research skills such as critical thinking, self-reflection and good academic practice are an integral part of teaching, learning and assessment activity. The development of personal learning and thinking skills, professionalism and employability skills are also fundamental to curriculum content and delivery.

To facilitate the independent learner, learning will mainly take place through small interactive group sessions where students are encouraged to explore content, ask questions, contribute to discussions and reflect on their progress. Emphasis is placed on individualised learning and avoiding teaching to the middle to ensure that the learning needs of each student are met. Learning will also take place to a lesser extent, through formal lectures. Each teaching session/lecture is supported by learning and assessment resources on the module VLE page on Brightspace. These include weekly objectives words to learn, slides/learning resources, interactive activities including assessments, useful videos and independent study. Every module will have at **least one** pre-identified element that will be taught through a flipped learning approach (screen cast lecture and associated assessment) in the independent study area on Brightspace.

For students with additional learning needs additional learning support plans are produced in collaboration with the University well-being and disability service and through these reasonable adjustments for teaching, learning and assessment are planned in advance for the duration of the student journey.

# OUTLINE SYLLABUS

* Use Case modelling, Use Case descriptions for requirements capture, Sequence Diagrams and system structure.
* Entities, attributes and relationships.
* Event / response modelling.
* Classes, fields and methods.
* Data modelling (primitive and user-defined types).
* Encapsulated types - introduction to objects, including inheritance, encapsulation, and class diagrams.
* Structured Programming (sequence, selection, and iteration).
* The basic components of algorithms described in a suitable language-independent notation: basic data structures – arrays, lists, trees; concepts for the modularisation of algorithms; spanning tree and shortest distance algorithms for graphs; basic searching and sorting algorithms
* Procedures, functions, parameter passing, variable scope.

# LEARNING OUTCOMES

## Knowledge and Understanding

On completion of this module students will be able to:

K1. Formulate a range of algorithms for manipulating data structures.

K2. Recognise the concept of correctness and verification and validation in testing computer programs.

K3. Describe the semantic concepts of a variety of abstract modelling techniques relevant to modern computing.

## Abilities

On completion of this module students will be able to:

A1. Apply abstract modelling techniques to a problem scenario to build models from a number of perspectives, in order to capture information and to devise appropriate event-driven and algorithmic solutions.

A2. Analyse problems and develop algorithms using a design approach (e.g. functional decomposition or object oriented design).

A3. Develop basic object oriented program, applying designed techniques and algorithms.

# ASSESSMENT STRATEGY

## Formative Assessment

Formative assessment will be provided through tutorial exercises and online tests.

Work will be set and feedback will be provided on the students' efforts. Model answers will be provided to certain exercises for comparison and evaluation. Students will be able to gain informal feedback on work in progress by taking it to the practical for their tutor to review. Students will also have the opportunity to self-asses their work to gain a greater understanding of the assessment process.

## Summative Assessment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Assessment | LO’s to be met | Weighting | Duration  (if timed teacher assessments) | Word count or equiv. if appropriate |
| Portfolio | K1 K2 K3 A1 | 50% |  |  |
| Coursework | K1 A2 A3 | 50% | **Final Assessment** | 30 hours |

Tutor Reassessment is available for portfolio component.

Neither assessment is marked anonymously.

**Assessment criteria**

In order to pass the programme students must achieve a minimum of 40% overall in the summative assessments.

A moderation activity will take place to ensure reliability in marks awarded.

Both coursework elements will be assessed using criterion-referencing. Each will be judged on the correctness, completeness and adherence to industry standards of:

* The UML models produced;
* Depth of insight in the critical evaluation.

**Appendix**

**Reading List**

Bell, A. (n.d.). Get Programming. Manning.

Bennett, S., Skelton, J., & Lunn, K. (2005). Schaum's outline of UML (2nd ed.). New York: McGraw-Hill.

Simon Bennett, Steve McRobb, and Ray Farmer. (2010). Object-oriented Systems Analysis and Design Using UML (4th ed.). McGraw-Hill.

Scott Chacon and Ben Straub. (2014). Pro Git (2nd ed.). Apress.

Deitel, P., Deitel, H., Mukherjee, S., & Bhattacharjee, A. (2014). C++ how to program. (Ninth edition / Paul Deitel, Harvey Deitel.). Boston: Pearson

Deitel, P., & Deitel, H. (2012). C++ how to program  (8th ed., International ed.,). Boston, [Mass.] ;: Prentice Hall.

Lunn, K. (2003). Software development with UML. Basingstoke: Palgrave.

.