MODULE **DESCRIPTOR**

MODULE TITLE	ADVANCED SOFTWARE MODELING		
MODULE CODE	CO3120 (L6)	**CREDIT VALUE**	20 UK CREDITS / 10 ECTS
SCHOOL	SCHOOL OF SCIENCES		

MODULE **AIMS**

1. To develop an understanding of formal development methods.
2. To develop skills and techniques to enable the student to use formal methods in the software development process.
3. To develop the student’s understanding of the theory of concurrent systems.
4. To develop skills and techniques in the implementation of complex software.

MODULE **CONTENT**

Indicative syllabus content:
The module takes a rigorous approach to software development. It examines the use of formal methods for the specification, development and verification of software. It presents a range of techniques for the analysis and implementation of real-time solutions. The student will investigate the theory and problems of concurrent systems.

Formal Methods of Software Development:
Introduction to a formal specification language (e.g. JML, VDM, Z or Spec#)
Design by Contract (Using a formal language to specify pre- and post-conditions)
Formal methods and UML: Object Constraint Language (OCL)
Evaluating methods for ensuring quality
Applications of discrete mathematics
Software standards and quality measures for high integrity software applications.
BNF and basic theory of grammars and parsing. Regular expressions (e.g. in scripting languages), Lexical analysis and use of parser generators.
Introduction to functional programming (e.g. Haskell or F#)

Metrics and Modelling:
Software metrics, Defining metrics: Goal, Question, Metric, scale type
The modelling process
Analytic modelling: finite state models and simple mathematical models
Introduction to system simulation

Development of Concurrent Software:
Problems of concurrency (e.g. communication and synchronisation, scheduling, non-determinism, mutual exclusion, deadlock, pseudo and true concurrency)
Real-time development methods: the design and implementation of real-time solutions
Features of real-time languages (e.g. tasks/threads, monitors, interrupt handling, semaphores)
Test-driven development

INTENDED **LEARNING OUTCOMES**

On successful completion of this module a student will be able to:

1.	Develop and interpret specifications in a formal notation.
2.	Use and evaluate models in software development.
3.	Analyse concurrency problems in a range of applications.
4.	Use appropriate tools and methods for concurrent software development.
5.	Design and implement a small concurrent system in an appropriate language.
6.	Evaluate software development techniques such as test-driven development, parser-generators.

TEACHING METHODS

Students will find this a difficult module because of the nature of the material. Spurious depth and abstraction will be avoided in favour of enhancing the student’s skills in developing and analysing formal specifications. Analogies will be drawn between programming and specification using model-based languages to help the students to apply their existing skills in developing and debugging programs to the development, testing and debugging of specifications. Where possible the teaching of formal specification in lectures will be example-based allowing discussion of alternative approaches. The need to ensure that the specification corresponds to the informal requirements will be stressed.

The examination will allow the students to demonstrate their knowledge and their understanding of the concepts. For example, a question might require the student to interpret part of a formal specification, or to identify errors in a faulty specification.

ASSESSMENT METHODS

This module is assessed through practical work (involving specification using formal and semi-formal methods, implementation of concurrent system and evaluation) and an examination.