MODULE **DESCRIPTOR**

MODULE TITLE	Advanced Software Engineering Techniques		
MODULE CODE	CO3401 (L6)	**CREDIT VALUE**	20 credits / 10 ECTS
SCHOOL	SCHOOL OF SCIENCE		

MODULE **AIMS**

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 software solutions. The student will investigate the theory and problems of concurrent systems.

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**

Formal methods of software development:

Introduction to a formal specification language (e.g. Code Contracts)
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.

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)
Concurrent features of modern 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 to develop a small concurrent system in an appropriate language.
5. Evaluate software development tools and techniques.

TEACHING METHODS

Students may 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.
Use will be made of libraries to integrate the process of developing a specification with the development of code, allowing the use of static analysis and run-time checking to verify the implementation against the specification.
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.

Tutorials will involve the development and analysis of specifications and discussion and development of sample code

Practical work will involve concurrent programming, the use of CASE tools for concurrent specification and design and the development of specifications using an appropriate tool. Practical programming techniques for concurrent programming in a modern programming language will be investigated and appropriate solutions developed. In teaching the module, the problems as well as the benefits of formal methods will be highlighted.

The assignment will provide the opportunity for the students to apply their skills in using formal methods in defining part of a problem and implementing a concurrent model of the entire system in an appropriate implementation language. Part of the assignment will require the student to evaluate their solutions to the given problem.

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 a Practical work involving specification using formal and semi-formal methods, implementation of concurrent system and evaluation (50%) and an examination (50%).