Module Code: **CMS 3405**

Module Title: **ADVANCED SOFTWARE DEVELOPMENT**

Schools involved in delivery:	Computing and Engineering
Name of Course(s):	MSc Advanced Computer Science
Module Leader:	Gary Allen
Location for delivery:	Queensgate, Department of Informatics
Module Type:	Core
Credit Rating:	15 credits
Level:	M - Masters
Learning Methods:	Lectures: 12 Tutorials/Practicals: 24 Unsupervised Study: 114
Pre-requisites:	None
Recommended Prior Study:	Skills in Procedural and Object Oriented Software Development
Co-requisites:	None
Professional Body Requirements:	Required for BCS Accreditation
Barred Combinations:	None
Graded or Non Graded:	Graded

Synopsis

This module provides learners with an opportunity to develop advanced skills in object oriented (OO) software design and development. The module begins with advanced OO concepts and semantics (polymorphism and genericity), moves on to techniques for real-time concurrent and distributed systems development (threads, semaphores, monitors, sockets, distributed objects architectures such as RMI and CORBA), examines contractual obligations and behaviour verification and validation techniques (pre- and post- conditions, invariants, assertions), and ends with an examination of contemporary advanced topics in OO software design (e.g. Design Patterns, Aspect Oriented Programming). In covering these topics, the module will examine the issues that software programmers and developers face every day in their quest to develop successful technology systems and applications.

Outline Syllabus

Advanced approaches to, and issues with:

- Inheritance and polymorphism.
- Genericity and portability in object-oriented software.
- Real Time and Concurrent Programming (e.g. threads, semaphores, monitors)
- Concurrent systems modelling and behaviour, e.g. concurrent state machines, message passing models and the role of formal event models such as Communicating Sequential Processes (CSP).
- Distributed and remote object access e.g. Java RMI and CORBA standards.
- Legacy code interfacing/wrappers.
- Contractual obligations, assertions, and behaviour verification and validation techniques (inc. QA and testing).
- Contemporary topics in OO software development, e.g. Design Patterns, AOP – and their importance to J2EE.

Learning Outcomes

1. Knowledge and Understanding Outcomes

On completion of this module learners will be able to:

1.1 Critically analyse a range of techniques for advanced OO programming.

1.2 Evaluate methods for the development of real-time, concurrent, and distributed software systems.

1.3 Evaluate advanced approaches to code verification and validation.
1.4 Expound current topics in advanced OO systems design.

2. Ability Outcomes

On completion of this module learners will be able to:

2.1 Design and implement complex data structures, using appropriate methods of encapsulation, inheritance, polymorphism, and genericity.
2.2 Model advanced real-time, concurrent, and distributed software systems as sets of objects, and defend the appropriateness and consistency of the model.
2.3 Implement such models using event-based and object-oriented techniques as appropriate.
2.4 Implement contractual obligations and apply appropriate verification and validation techniques.
2.5 Implement, with originality where appropriate, the findings from research for LO 1.4 in software development projects.

Assessment Strategy

Formative assessment

Formative assessment will be via tutorial/practical exercises designed to guide the students' learning, and to prepare them for the summative assessment. In addition, revision sessions will use previous tests to prepare learners for the online tests, and formative feedback will be provided on the students' answers to the questions.

Summative Assessment

Assessment tasks (including assessment weightings)

The module will be assessed by a software development assignment (50%) and a series of online tests (50%).

The assignment will involve the design, implementation, and evaluation of a software system, including real time, concurrent and/or distributed requirements. The students will be required to defend their chosen design and to implement appropriate 'design by contract' assertions in the code. The assignment will assess all ability outcomes.

The tests will examine all knowledge outcomes.

Assessment Criteria

The assignment will be assessed on the appropriateness of the chosen design (including data structures, class hierarchy, and implementation techniques), the completeness and correctness of the solution (evidenced by the testing and evaluation), and the quality of the critique defending the design decisions made.

The tests will be marked in accordance with normal university practice and the answer scheme produced by the module leader.

Learning Strategy

The lectures will be used to deliver the key background material, and to demonstrate the application of the methods and techniques addressed within the module. This will be supported by tutorials and practicals designed to allow the concepts and techniques to be put into practice through the use of exercises and case studies. Learners will be expected to carry out independent reading in support of the lecture programme and before/after the block delivery of the module.

Appendix: Indicative References

Journals: Journal of Information Technology, IEEE Software, Computing, Computer Weekly

Books:

Deitel, Deitel, & Santry, (2004) *Advanced Java 2 Platform – How to Program* Prentice-Hall

Darrel Ince (2000) *From Data Structures to Patterns* MacMillan Press

K. Czarnecki, U. Eisenegger (2000) *Generative Programming: Methods, Tools and Applications* Addison-Wesley

Cetus Links (definitive list of web links for all aspects of OO programming): http://www.cetus-links.org/

For background reading (e-books on Books 24x7):

Faison, T (2006) *Event-Based Programming: Taking Events to the Limit,* Apress

Richardson, W.C. et al. (2007) *Professional Java JDK (latest) Edition*, Wrox Press