

Agile Software Development

Learning Outcomes:

On successful completion of this module, the student should be able to:

- 1 Demonstrate an understanding of agile development methodologies.
- 2 Demonstrate an understanding of the range of tools available to support the software development process, particularly agile development techniques.
- 3 Demonstrate an ability to manage a modern medium scale software development project with respect to configuration management and deployment of standardised software project policies through the use of auto building software.
- 4 Demonstrate an understanding of the principles behind the tools used so as to be able to learn new tools as these evolve.
- 5 Use agile methods to improve the effectiveness of their own software development processes.
- 6 Implement a simple client/server application using standard socket-based APIs. Demonstrate Test Driven Development techniques in this context.

Syllabus Content:

This module will address a subset of the tools and technologies required to support the development of reliable, efficient and scalable software services. The focus is on use of Agile Development methods requiring test-driven developed and regular automated software builds. The aim is to assemble a toolkit of modern tools that enable the set-up of a software development process where this structure is automated by the tools. This module is designed to be very practical serving as a support to software development for the dissertation.

Pre-requisites:

- 1 Strong programming skills with Java experience (graduate of a BSc(Hons) or BEng involving significant programming experience)
- 2 Object-Oriented Programming and Design skills. (BSc(Hons) level Systems Analysis)
- 3 Moderate understanding of computer architecture and of operating systems.
- 4 Moderate understanding of distributed computing

Indicative syllabus content:

1. Agile Development in Java

- 1.1. Introduction to Agile Development
- 1.2. Java Review
- 1.2. Agile Practices

2. Test Driven Development

- 2.1. Principles of testing.
- 2.2. Unit testing.
- 2.3. Integration tests.
- 2.4. Acceptance tests.
- 2.5. Performance tests.

3. Configuration Management

- 3.1. Principles configuration management
- 3.2. Principles of build management
- 3.2 First generation systems, e.g. CVS, make.
- 3.3. Second generation tools, e.g. Subversion, ant, maven
- 3.4 Continuous Integration, e.g. Cruise Control.

4. Network Programming

- 4.1 Sockets.

4.2 Threads.

4.3 Streams and Channels.

4.4 XML Parsing.

Practical Programme:

The practical programme will involve a 3 assignments based on implementing a set of open source tools for supporting software development which us developed using Agile Development Techniques.

The assignments may be integrated with a software assignment for another module and/or the dissertation.

Communications Infrastructure & Security

Learning Outcomes:

On successful completion of this module, the student should be able to:

1. describe how the latest technologies in Local Area and Wide Area Networks (protocols, architectures) operate and how they can be deployed
2. compare/evaluate major networking technologies found on the Internet today and assess their suitability for different scenarios and user requirements.
3. demonstrate a systematic knowledge and understanding of advanced routing and switching concepts
4. demonstrate an understanding and knowledge of implementation of wireless networks
5. discuss the key aspects of OS and network security in a heterogeneous environment.
6. analyse security approaches for leading applications like e-mail and the Web.
7. make effective use of leading network security protocols and applications.

Syllabus Content:

This module will build on the student's prior knowledge of computer networks and provide the student with a thorough understanding of many of the latest LAN and WAN technologies. The main areas covered in this module are TCP/IP, Routing, Switching and Wireless communications, as well as security concepts and technologies and their applications. The practical component of the module will allow the student implement and test many of the concepts covered in the lectures.

Pre-requisites:

- Basic understanding of Networks and TCP/IP.
- Basic understanding of Internet Protocols.
- Moderate programming skills (Java or C/C++).
- Moderate mathematical ability (at least one year of undergraduate mathematics).

Indicative syllabus content:

1. In depth treatment (assuming prior knowledge) of Ethernet, IP, UDP, TCP; use of protocol analyser software to capture and analyse different types of frames;
2. IP subnetting, VLSM, CIDR
3. IPv6 Protocols, Addressing, configuration.
4. LAN switching, RSTP, VLANs, VTP
5. Routing – algorithms and protocols; autonomous systems, BGP, RIP, OSPF, example router configuration using Cisco routers;
6. Wireless Protocol architectures (OSI layer 2): Bluetooth, 802.11.
7. Emerging Communications technologies - Mobile IP, Sensor networks
8. Security Services & Policy: vulnerabilities and attacks, security assessment
9. Cryptography: conventional encryption, public key cryptography, authentication and hash functions, key management, certificates and trust management
10. Communications & Network Security: IP security, VPNs; Transport Layer security; Authentication protocols; Security in wireless networks
11. System security: firewalls; intrusion detection & honeypots; passwords
12. (In)secure code: Secure software development; good & bad programming practice; Malicious software

Practical Programme:

This module adopts a hands-on practical approach. The aim of this module is to allow the student to develop a working knowledge and understanding of how communications infrastructures and network protocols operate.

1. Communications Infrastructure
 - a. Protocol Analyser
 - b. Routing – Static, RIP, OSPF

- c. Switching – VLANs, VTP
 - d. Wireless
- 2. Network Security
 - a. Exploring host security
 - b. Cryptographic tools – e.g. PGP, OpenSSL, key management
 - c. Secure coding
- 3. Research assignment

Design Patterns

Learning Outcomes:

On successful completion of this module, the student should be able to:

1. Understand core design principles and be able to assess the quality of a design with respect to these principles.
2. Be capable of applying these principles in the design of object oriented systems.
3. Demonstrate an understanding of a range of design patterns. Be capable of comprehending a design presented using this vocabulary.
4. Be able to select and apply suitable patterns in specific contexts. Be able to critically analyse these applications and assess tradeoffs associated with pattern implementations
5. Understand and apply refactoring techniques in the context of design patterns.
6. Understand the broader scope addressed by Architectural Styles. Be capable of relating design patterns to these styles.

Syllabus Content:

Building on an undergraduate-level software development knowledge base, the central focus of the module is to broaden the “design vocabulary” of the student to incorporate best practice in object oriented software development.

The module will revisit core design principles and frame these in the context of design patterns. A set of patterns are examined in detail, both in isolation and in the context of integrated applications. Particular attention is paid to relating patterns and assessing design tradeoffs. Alternative Pattern classifications are examined. The role of refactoring in is examined and pattern-based refactoring techniques are presented. Broader software architectural issues are explored and attention is given to relating the scope of architectural styles to the design pattern literature.

Pre-requisites:

- *Agile Software Development (from this programme).*
- or*
- *Basic understanding of computer architecture and of operating systems.*
- *Programming experience in an Object Oriented Programming Language (graduate of a Honours BSc or BEng involving significant programming experience).*
- *Some exposure to software modelling concepts and notations.*
- *Experience of Test Driven Development.*

Indicative syllabus content:

1 Principles

- 1.1 Types & Classes
Fragile Base Classes, Programming to Interfaces
- 1.2 Design Principles
Single Responsibility, Open Closed, Liskov Substitution, Dependency Inversion
- 1.3 Packaging Principles
Acyclic Dependencies
- 1.4 Refactoring
Code smells, named refactorings & the role of unit testing

2 Design Patterns

- 2.1 Creational Patterns
Singleton, Factory, Prototype
- 2.2 Behavioural Patterns
Command, Observer, Strategy, Template Method, State, Iterator, Chain of Responsibility
- 2.3 Structural Patterns:
Façade, Proxy, Bridge, Composite, Adapter
- 2.4 Pattern Based Refactoring

Refactoring towards & away from specific patterns

3 **Architectural Styles**

- 3.1 Data Flow, Replication
- 3.2 Hierarchical, Peer-to-peer
- 3.3 Mobile code

Practical Programme:

This module has a strong practical element. The focus of this element is on the application of a subset of the design patterns presented in the curriculum, with particular attention paid to the interaction of multiple patterns in a single system. Additionally, best practice in agile development will be assumed & inspected in submitted assignments (particularly unit tests and code style issues).

Students will be required to submit three completed assignments. The first assignment is to be presented by the students early in the module, delivering a base-line feature set realised by the students independent of a pattern vocabulary. This is re-implemented in two subsequent assignments, introducing patterns in isolation and in combination. Sample solutions to these assignments are a core pedagogical tool in presenting and exploring patterns. Additionally, patterns encountered in standard platform components (JDK and other) will be an integral part of the assignments.

Communications Services Management

Learning Outcomes:

On successful completion of this module, the student should be able to:

1. Demonstrate an understanding of fixed, mobile and Internet service platform technologies
2. Apply Utility Computing concepts and technologies in the design and development of software services.
3. Design and develop software converged network services utilising state-of-the-art service creation tools and frameworks.
4. Understand the foundations in terms of paradigms, basic concepts and strategies in network analysis and network management.
5. Appreciate the traditional management paradigm and its adaptation for today's service and system management

Syllabus Content:

This module will address a subset of the practices and technologies required to develop reliable, efficient and scalable software services which span multiple communications networks. The student is introduced to the fundamentals of classic network and service management as well as to new industrial approaches that enable him/her to understand administration and maintenance of the related infrastructure. A key goal will be to familiarise the student with the concept of integrated management approaches interrelating several strategies for specific management tasks.

Pre-requisites:

Basic programming skills with some Java experience (graduate of a BSc or BEng involving significant programming experience.) Object-Oriented Programming and Design skills. (BSc. level Systems Analysis.) Moderate understanding of computer architecture and of operating systems.

Indicative syllabus content:

Converged Communications

SIP as a signalling protocol for Internet-based applications and for 3G services. The Session Initiation Protocol (SIP) is a signalling protocol used for establishing sessions in an IP network. A session could be a simple two-way telephone call or it could be a collaborative multi-media conference session. The ability to establish these sessions means that a host of innovative services become possible, such as voice-enriched e-commerce, web page click-to-dial, and Instant Messaging with buddy lists.

The Extensible Messaging and Presence Protocol (XMPP): IETF's formalisation of the base XML streaming protocols for instant messaging and presence (developed within the Jabber community).

Traditional Management Architectures

In this part of the module, the student will study traditional management structures for data and telephony networks. This can include SNMP (v1,2 and 3) with respect to data comms, TMN and TINA with respect to telephony and OSI X.500 or X.700 for both.

Emerging Management Architectures

Students will gain exposure to emerging management architectures, with a strong focus on fixed-mobile convergence. Students will build a software-based testbed to run experiments using these technologies, which may include Policy Based Management, IP Multimedia Subsystem (IMS) or DEN-NG.

Practical Programme:

The practical programme will involve 3 assignments: 1 discursive essay and 2 software-based development project.

Research Methods

Learning Outcomes:

On successful completion of this module, the student should be able to:

1. Provide an overview of the research process
2. State clearly their research problem and associated research questions arising, including both descriptive and either explanatory or exploratory questions
3. Conduct a preliminary literature review of the concepts comprising the research questions
4. Set out clearly a series of theoretical propositions for testing and demonstrate clearly how they arise from the literature review
5. Set out the main elements of a potential research instrument for testing the hypotheses, including a critical and comparative analysis of the proposed instrument
6. Understand how to classify and present data associated with hypothesis testing
7. Understand how to interpret data gathered to test proposed theory
8. Set out limits and implications of a research study in preliminary form
9. Distinguish between quantitative and qualitative approaches and methods
10. Prepare a mini-dissertation research proposal
11. Prepare a research plan

Syllabus Content:

This module aims to formally induct the MSc students into the research process including theory and provide an overview of methodologies and methods associated with carrying out independent research. This module is designed to provide a basic understanding of the scientific research process.

Pre-requisites:

Basic mathematical skills.

Indicative syllabus content:

1. Introduction to the research process: overview of the classical scientific research process, becoming a postgraduate researcher, reflexivity, personal and research objectives
2. Topic selection, problem formulation, setting out research study objectives, formulating different types of research questions (descriptive, exploratory, explanatory, emancipatory etc.), qualities of focused RQs
3. Critiquing the academic literature: features and sources of academic publications in Computing and IS, preparing a focussed literature survey and review, referencing styles and conventions, the relationship between the literature review and research questions, the relationship between the literature review and theory, reviewing criteria for academic publications
4. Theory formulation: concepts & variables, propositions and hypotheses, different types of propositions, setting out testable propositions, relationships between propositions (building theoretical frameworks)
5. Designing a research approach to test theory: fixed vs. flexible designs, Overview of Quantitative, Qualitative and Multi-method approaches
 - 5.1. Quantitative methodological approaches
 - 5.1.1. The principles of quantitative research
 - 5.1.2. Experimental research design
 - 5.1.3. Sampling and statistics
 - 5.1.4. Questionnaire design
 - 5.1.5. Types of variable

- 5.1.6. Methods for one variable
- 5.1.7. Methods for 2 variables
- 5.1.8. Outline of multivariate methods
- 5.1.9. Introduction to statistical inference
- 5.1.10. Computer applications – SPSS & Minitab
- 5.1.11. SPSS and Minitab.
- 5.2. Overview of Qualitative methodological approaches
 - 5.2.1. Theory and its relation to qualitative research methods
 - 5.2.2. Grounded theory
 - 5.2.3. Ethnography
 - 5.2.4. Action research
 - 5.2.5. Case study research
 - 5.2.6. The interview as an investigative method
 - 5.2.7. Observation and the nature of observation
 - 5.2.8. Reflection and reflexivity within qualitative research
 - 5.2.9. Techniques and issues in gathering data
 - 5.2.10. Principles of data analysis within qualitative research
 - 5.2.11. Interpreting qualitative data
- 6. Qualities of a research proposal: Criteria of coherence, comprehensiveness and currency, originality, impact, validity, reliability, generalisability, significance, rigour, some epistemological considerations
- 7. Conducting a Research Study
 - 7.1. Overview of preparing a thesis/ research paper for submission under Taught M.Sc. guidelines
 - 7.2. Major sections of a research publication and their purpose
 - 7.3. Evaluation and reviewer criteria for research publications
 - 7.4. Managing the Supervisor – the role of the supervisor and the role of the student
 - 7.5. Setting research agenda
 - 7.6. Research journals and diaries
 - 7.7. Setting out and managing a research plan
- 8. Conducting research in WIT: facilities and programmes

Recommended Learning Modes

Lectures, group work, practical sessions and workshops.

Practical Programme:

Seminar and workshop sessions e.g. exchange of ideas, methods workshops, computer packages for research e.g. SPSS & Minitab, NU-Dis, online resources.

Graph Theory and Optimisation

Learning Outcomes:

On successful completion of this module, the student should be able to:

1. Demonstrate an understanding of the general notion of complexity classes, P and NP, completeness and hardness, and the relationships between classes by reduction.
2. Demonstrate an understanding of the theory underlying exact and heuristic approaches to combinatorial optimisation problems.
3. Prove and interpret standard results in graph theory.
4. Develop, implement, and critically evaluate the correctness and performance of standard graph and combinatorial optimisation algorithms.
5. Select and implement appropriate formulations and algorithms from graph theory and combinatorial optimisation to analyse problems from computing, engineering, and operations research.

Syllabus Content:

The module will introduce the students to the fundamental concepts and techniques in graph theory and network based combinatorial optimisation, focusing on the relationships between algorithms and associated data structures. The module also includes discussions of applications to problems from computing, engineering, and operations research thus illustrating the broad applicability of the theory.

Pre-requisites:

- *Basic programming skills (graduate of a BSc or BEng involving significant programming experience)*
- *Moderate mathematical ability (at least one year of undergraduate mathematics), in particular, basic linear algebra (including eigenvalues and eigenvectors) and an understanding of simple combinatorial counting processes involving permutations and combinations.*

Indicative syllabus content:

1. Algorithmic Complexity

Order notation (O , Θ). Analysis of algorithms. Greedy heuristics and approximate algorithms. Classification of decision problems: complexity classes of P, NP and Co-NP. NP-completeness. NP-hardness.

2. General Graph Theory

Core concepts and definitions: connectivity, dimension, graph traversal. Special graphs and their properties: complete, regular, interval vertex-transitive, and circulant graphs. Directed graphs and weighted graphs.

3. Optimal Routes in Graphs

Graph traversal: DFS, BFS and hybrid traversal. Applications of shortest routes in graphs: determination of shortest paths in non-negatively weighted graphs via the methods of Dijkstra and of Floyd. Estimation of the algorithmic complexities of Dijkstra's method & Floyd's method. Applications of longest routes in directed acyclic graphs.

4. Minimum Spanning Trees

Applications of trees and of rooted trees. Properties of rooted trees and of m -ary trees (with binary trees as a special case). Determination of minimum spanning trees by Kruskal's, Prim's and Boruvka's method. Analysis of MST algorithms.

5. Planarity

Euler's formula. Applications of graph planarity with respect to optimal facility lay-outs. Kuratowski's theorem. Crossing numbers of graphs embedded in the plane.

6. Centres and Medians

Vertex eccentricity; radius and diameter of a graph. Graph centre and a graph median. Applications to optimal facility placement in communication networks.

7. Vertex and Edge Colourings

Vertex and edge chromatic numbers. Heuristic methods for graph colourings. Applications of vertex & edge colourings with respect to scheduling.

8. Eulerian Graphs

The Königsberg bridge problem. Chinese postman problem and variants. DeBruijn sequences and gray codes. Application to model based testing.

9. Hamiltonian Graphs

The travelling salesman problem (TSP): heuristics for the travelling salesman problem (such as the method of nearest neighbours and Christofides' method of cheapest insertion). Post-optimisation methods for the travelling salesman problem (such as 2-opt).

10. Network Flow

The Matching problem. Matching and augmenting paths. The network flow problem. Assignment problem. Weighted Edge cover problem. Multi-commodity flow problem.

11. Emerging Topics

Dynamic graph algorithms: clustering and topology trees, sparsification. Randomised algorithms. Graph drawing algorithms. On-line algorithms: paging scheduling and load balancing.

Practical Programme:

The practical component will involve studying a number of problems and adapt algorithms from the module into solving these problems, as well as a practical implementation of the relevant data-structures and algorithms. This way the students get practical, hands-on experience with techniques from graph theory and combinatorial optimisation. The ability to apply the theory covered in this module in terms of progressing from a problem, to selecting and implementing an appropriate solution strategy, and ultimately interpreting the results is a fundamental learning objective of this module. This ability is assessed primarily through the practical component.

Dynamic User Interface Development

Learning Outcomes:

On successful completion of this module, the student should be able to:

- 1 General competency in Dynamic User Interface Development.
- 2 Understand the difference between Simple and Rich Internet Applications.
- 3 Understand the benefits of Rich Internet Applications at both technical and business levels and identify applications that are best suited for RIAs.
- 4 Identify and compare existing technologies enabling the development of Rich Internet Applications based on characteristics such as ubiquity, industrial strength and usability.
- 5 Be familiar with the evaluation of user interfaces according to usability principles.
- 6 Be familiar with the implementation of user interfaces, understand the implication of human cognition on user interface design and distinguish between good and bad user interface.
- 7 Use object-oriented languages for the design of Dynamic User Interfaces
- 8 Be able to reuse and create libraries for the design of dynamic interfaces
- 9 Be capable of designing and delivering a Rich Internet Application.

Syllabus Content:

Building on an undergraduate-level software development knowledge base, the central focus of the module is to enable the understanding and critical evaluation of highly interactive User Interfaces. This module will build on the previous module “Dynamically Types Programming” where students have acquired a sound understanding of Object Oriented Programming. It will allow to develop Rich Internet Applications (RIA) with an emphasis on scalability and usability. It will also draw a parallel between existing technologies.

Pre-requisites:

- Basic understanding of computer architecture and of operating systems.
- Programming experience in an Object Oriented Programming Language (graduate of an Honours BSc or BEng involving significant programming experience).

Indicative syllabus content:

1. Principles

- 1.1 Introduction to Dynamic UIs
 - RIAs Characteristics
 - Benefits of RIA
 - Overview of Available Technologies for RIAs
- 1.2 Creating Usable Interfaces
 - Introduction to Usability Principles
 - Applying Usability Principles to Dynamic User Interfaces

2. Development of Dynamic User Interfaces

- 2.1 Using JavaScript/AJAX
 - DOM, XML and CSS
 - Event handling
- 2.2 Using Java
 - Java FX
 - Java Applets
- 2.3 Using Dedicated IDEs
 - Overview of Flash and Flex
 - Scripting
 - Built-in libraries

3. *Technology Integration*

Design, develop and deploy a commercial application using a combination of RIA technologies.

Practical Programme:

This module has a strong emphasis on practical applications. Lectures will be used to introduce new topics and their related concepts. Computer based practicals will be used to illustrate principles introduced in the lectures.

Best practice in Object Oriented Development and Interface Usability will be introduced and tested in the submitted assignments.

Students will be required to submit two completed assignments. The first assignment is to be presented by the students early in the module, showing a good understanding of the design of Dynamic user Interfaces.

The second assignment will expect students to develop a commercial solution implementing rich content.

Relational Persistence

Learning Outcomes:

On successful completion of this module, the student should be able to:

- 1 Demonstrate competency in designing, implementing and manipulation a relational database schema for a rich problem domain
- 2 Be able to explain the issues and solutions to maintaining its integrity in a multi-user environment.
- 3 Understand the role of XML as a framework for defining domain specific languages and be able to define simple schemas using both DTD and XML Schema. Appreciate the application of this technology to the object relational persistence domain.
- 4 Demonstrate an understanding of a subset of tools for processing XML documents, including persisting, parsing, querying and transforming.
- 5 Understand the object-relational paradigm mismatch and be able to demonstrate competency in applying the mapping concepts and strategies provided by a best-of-breed ORM framework to a domain model
- 6 Demonstrate an understanding of an ORM framework's support for conversational object processing, including querying, fetching strategies, caching, business transactions.

Syllabus Content:

This module is focused on persistence strategies appropriate for an object-oriented software paradigm. As relational databases dominate this technology space, the module's first section will be a condensed look at its fundamentals from both a designer and application developer perspective. From an OO application developer perspective, the student will get hands-on experience in programming the abstractions layer provided by a leading open source solution to the object/relational paradigm mismatch. A third section covers XML, as an enabler of ORM, an alternative persistence strategy and more generally an information structuring technology.

Indicative syllabus content:

1. Relational Databases
 - Design
 - Querying
 - Transactions and Concurrency
2. XML
 - Schema language definition – DTD; XML Schema
 - Querying – XQuery; XPath
 - Document transformation
 - Parsers
3. Object Relational Mapping
 - The paradigm mismatch
 - Mapping - Entities, Collections, Associations, Inheritance
 - Entity management.- querying, persistence lifecycle,
 - Performance – transitive persistence, fetching strategies; caching,

Practical Programme:

The practical programme will involve 3 assignments based on the three main syllabus areas, SQL, XML & Object Relational Mapping.

The assignments may be integrated with a software assignment for another module and/or the dissertation.

Enterprise Component Development

Learning Outcomes:

On successful completion of this module, the student should be able to:

1. Comprehend the architecture of a multi-layered enterprise application and the disadvantages associated with traditional approaches to accessing infrastructure services.
2. Describe the key concepts underpinning lightweight enterprise application frameworks (AOP, IOC, and Declarative service application) and how they benefit application architecture – coupling, modularity, testability, simplicity.
3. Explain how IOC is realised in a best-of-breed container and comprehend the full extent of its power in the management and configuration of an application's components, including life cycle management, externalising deployment configuration, internationalisation, and event management.
4. Comprehensively define the AOP model and utilize an implementation of this model to illustrate the enrichment of objects.
5. Demonstrate the exploitation of the key concepts to achieve declarative application of infrastructure services to an enterprise application's components.
6. Implement and deploy a web application that demonstrates MVC Model2 as well as Web 2.0 characteristics
7. Combine all of the above outcomes to develop a skeleton multilayered enterprise application requiring a range of infrastructure services, using the tools and techniques of Agile development.

Syllabus Content:

This module will address the use of lightweight application frameworks in the development of secure, transactional, web-enabled, multi-tiered enterprise applications. This technologies' non-invasive characteristic results in greatly simplified business logic code consisting of so-called POJOs (plain old java objects). The foundation concepts associated with this technology are explained followed by an in-depth look at the leading open source products in this market. There will be a strong emphasis on the practical side, where the student will experience the benefits of the frameworks toward testability, reuse, maintainability, and ease of configuration. The skills and knowledge acquired from earlier modules - Agile Software Development and Relational Persistence - will be consolidated, giving a more complete picture of enterprise application development.

Pre-requisites:

1. *Strong programming skills with Java experience (graduate of a BSc(Hons) or BEng involving significant programming experience)*
2. *Object-Oriented Programming and Design skills. (BSc(Hons) level Systems Analysis)*
3. *Moderate understanding of computer architecture and of operating systems.*
4. *Moderate understanding of distributed computing*

Indicative syllabus content:

a. Background

Enterprise application layers; Code bloat ; Code scatter and tangling; Dependency management; Infrastructure services ; Programming best practice.

b. Core IOC Container

Declarative Component configuration; Bean lifecycle management; Internationalisation;

c. The AOP paradigm

Concepts; Framework support.

d. Middle-tier infrastructure services

Security; Remoting; Transaction management; Persistence; Messaging; Scheduling. Declarative application

e. The web tier

MVC Model 2; Web Conversations.

f. Agile development support

Testing; Deployment.

Practical Programme:

This module has a strong practical element. Students are required to demonstrate competency in the use of the frameworks discussed in the lectures. Additionally, best practice in agile development will be assumed and inspected in assignment work submitted – testing, system build. The student will be required to develop a multi-tiered secure, transactional enterprise application with a manageable set of user features. The problem domain can be of the student's choosing or taken from an application being developed for another module/dissertation. It will be submitted in three phases, each one demonstrating the framework features covered to-date.

Ubiquitous and Pervasive Computing

Learning Outcomes:

On successful completion of this module, the student should be able to:

1. Demonstrate an understanding of technologies enabling ubiquitous and pervasive computing; in particular in networking, processing, and positioning.
2. Recognise the suitability of software structures dependent on the addressed hardware requirements
3. Analyse the technological requirements of current problems in ubiqcomp,
4. Identify the maturity of the technology required for a particular problem
5. Assess the feasibility to market a certain product in comparison to the time available for its development.

Syllabus Content:

In this module, the interdisciplinarity of the approaches in Ubiquitous and Pervasive Computing will be addressed. The red line is considered to span from technological prerequisites, over software and hardware required, to applications.

The view will be enriched by the consideration of ongoing research and development in this area on an international level.

Pre-requisites:

1. *Basic programming skills (graduate of a BSc or BEng involving significant programming experience)*
2. *Moderate understanding of computer networking*
3. *Moderate understanding of computer architecture and of operating systems.*
4. *Moderate understanding of distributed computing*

Indicative syllabus content:

- 1. Overview and driving factors of ubiquitous and pervasive computing**
- 2. Enabling Technologies**
 - 2.1. Wired communication technology (Infrastructure network, Audio/Video networks, Internet Protocol, etc.)
 - 2.2. Wireless and mobile communication technology (WLAN, Bluetooth, wireless control networks, sensor communication, etc.),
 - 2.3. Ad hoc networking
 - 2.4. Positioning Technology (indoor vs. outdoor, GPS, GSM-positioning, infrared, ultrasonic, visual, dead reckoning, hybrid)
- 3. Context Awareness, Autonomous Computing**
 - 3.1. Wireless Sensor Networks
 - 3.2. Sensors and Actuators
 - 3.3. Computational Perception
- 4. Software infrastructures**
 - 4.1. Universal Plug and Play
 - 4.2. Home Audio Video Interoperability
 - 4.3. Super Distributed Objects
 - 4.4. Middleware issues
- 5. Power Supply for ubiquitous devices**
 - 5.1. Batteries
 - 5.2. Energy scavenging from environment
 - 5.3. Human body as energy supply
- 6. Application Areas**
 - 6.1. New Materials
 - 6.2. Small Artefacts
 - 6.3. Networked Embedded Systems
 - 6.4. Vehicular communication (communication with vehicles, communication between vehicles)

- 6.5. User- or Individual-centric Networking
- 6.6. "Smart" Appliances, Ambient Intelligence, User interfaces
Applications, Prototypes,
- 6.7. Emerging scenarios in research and business
- 7. **Social, Privacy and Security impacts**

Practical Programme:

The heterogeneity of the module requires focusing the practical assignment on individually selected topics.

The practical approach in this module will require students to prepare individual presentations of 30 to 60 min based on selected research papers/issues.

Service Oriented Architecture

Learning Outcomes:

On successful completion of this module, the student should be able to:

1. Demonstrate an understanding of software oriented architectures.
2. Demonstrate an understanding of the service composition..
3. Demonstrate an ability to manage a modern medium scale software development project using SOA principles.
4. Demonstrate an understanding of the principles linking business processes, process oriented architectures and service oriented architectures.
5. Demonstrate and ability to implement a service oriented application.

Syllabus Content:

This module will address the use of a service oriented approach to construct a software architecture. The module will introduce the principles of service oriented architectures. The module will and analysis and design approach which enables a large scale business process to be automated using a service oriented approach.

Pre-requisites:

1. *Strong programming skills with Java experience (graduate of a BSc(Hons) or BEng involving significant programming experience)*
2. *Object-Oriented Programming and Design skills. (BSc(Hons) level Systems Analysis)*
3. *Moderate understanding of computer architecture and of operating systems.*
4. *Moderate understanding of distributed computing*

Indicative syllabus content:

1. SOA and Web Services Fundamentals

- 1.1 Introduction to SOA
- 1.2 The Evolution of SOA
- 1.3 Web Services and Primitive SOA

2. SOA and WS-* Extensions

- 2.1. Activity Management and Composition.
- 2.2. Messaging, Metadata, and Security.

3. SOA and Service Orientation

- 3.1. Process Oriented Architecture
- 3.2. Principles of Service-Oriented
- 3.3 Service Layers

4. Building SOA – Planning and Analysis

- 4.1 SOA Delivery Strategies.
- 4.2 Service-Oriented Analysis.
- 4.3 Service-Oriented Modelling

5. Building SOA – Technology and Design

- 5.1 Service-Oriented Design.
- 5.2 SOA Composition Guidelines
- 5.3 Service Design
- 5.4 Business Process Design
- 5.5 Fundamental WS-* Extensions

Practical Programme:

The practical programme will involve a set of 3 assignments based on developing and implementing a set of Web services to support intra- or inter-organisational business processes. The assignments may be integrated with a software assignment for another module and / or the dissertation.

Dynamic Languages

Learning Outcomes:

On successful completion of this module, the student should be able to:

1. Generalize competency in dynamic languages to any programming domain, and classify domains by level of support for dynamic features.
2. Apply the features of dynamic languages to practical problems. Be capable of designing and delivering complete solutions in a dynamic language.
3. Demonstrate an understanding of the prototype-based object model. Be able to modify and adapt a traditional object hierarchy into a prototype model.
4. Formulate an effective quality approach for dynamic systems. Understand the impact of the loss of static type-checking.
5. Understand the strong/weak, static/dynamic typing distinctions, be capable of generating examples and critically assessing the inherent trade-offs.
6. Describe functional programming and connect the features of pure functional languages to dynamic languages.

Syllabus Content:

The core aim of the module is to expand the problem-solving abilities of the student by moving beyond traditional object-oriented algorithms and design patterns.

The module will introduce two dynamic languages, Ruby and JavaScript, and use these to develop an appreciation of dynamic language techniques, structures and idioms. The core concepts of functional programming will also be introduced in order to unify the material in the module. Comparison to structures and patterns in Java will be used to illustrate these concepts.

Particular emphasis will be placed upon understanding the trade-offs inherent in the use of dynamic languages, particularly with respect to performance, quality assessment and code accessibility.

Finally, design and delivery of complete, commercial-grade systems using domain-oriented frameworks (Ruby-on-Rails, Prototype) will form the basis for continuous assessment.

Pre-requisites:

1. *Basic understanding of computer architecture and of operating systems.*
2. *Programming experience in an Object Oriented Programming Language (graduate of a Honours BSc or BEng involving significant programming experience).*
3. *Some exposure to software modelling concepts and notations.*

Indicative syllabus content:

1. ***Principles***
 - a. Object Ontologies
Classes, prototypes, and functions.
 - b. Type Patterns
Static and dynamic, strong and weak
 - c. Dynamic Idioms
Higher-order functions,
recursion, list comprehension, closure, currying.
 - d. Functional Programming
Overview of paradigm, LISP-style syntax, and lambda calculus.
2. ***Language Competency***
 - a. Environment
Toolset and IDE use, project organisation, debugging, deployment, virtual machine hosting.
 - b. Community
Coding standards, online resources, thought-leader blogs.
 - c. Development

Architectural approaches, unit testing strategies, performance considerations, refactoring from traditional object hierarchies.

d. Functional techniques

Application of common functional idioms and techniques using dynamic code.

3. Domain Frameworks

a. *Ruby-on-Rails*

Design, develop and deploy commercial-grade applications.

b. *Prototype*

Apply and integrate into design architecture and working solutions.

Practical Programme:

This module is directly practical, despite the theoretical nature of the core material. The focus is on the utility of the techniques and ideas covered, and their usefulness in modern large-scale software development. Practical activities will be motivated by applicability to common problem spaces, particularly in distributed systems. Students will also be encouraged to adopt an agile approach to development.

Assessment shall consist of three assignments. The first assignment is to be presented by the students early in the module, in order to demonstrate familiarity with key dynamic language concepts, implemented in the Ruby language. The second (and largest) assignment will cover the Ruby-on-Rails framework, and students are to deliver a full end-to-end system, demonstrating effective use of dynamic techniques. The third and final assignment will cover the more theoretical aspects of the module, using the JavaScript language in an AJAX context.

Mobile Application Development

Learning Outcomes:

On successful completion of this module, the student should be able to:

1. Generalise competency in Mobile Development.
2. Understand the difference between Desktop and Mobile Applications and its implications.
3. Understand the benefits of Mobile Applications at both technical and business levels and identify applications that are best suited for mobile phones.
4. Identify and compare existing technologies enabling the development of Mobile Applications
5. Be familiar with the creation of effective user interfaces for mobile phones.
6. Be familiar with the development workflow of mobile applications,
7. Understand the structures (software and hardware) underpinning the design of mobile applications.
8. Use object-oriented languages for the design of Mobile Applications
9. Be able to reuse libraries for the design of mobile applications
10. Be capable of designing and delivering a Mobile Application.

Syllabus Content:

Building on an undergraduate-level software development knowledge base, the central focus of the module is to enable the understanding and critical evaluation of Mobile Applications.

This module will allow to develop Mobile Applications with an emphasis on Java and Symbian devices. It will also draw a parallel between existing technologies.

Pre-requisites:

- *Basic understanding of computer architecture and of operating systems.*
- *Programming experience in an Object Oriented Programming Language (graduate of a Honours BSc or BEng involving significant programming experience).*

Indicative syllabus content:

1 Principles and Design

- 1.1 Introduction to Mobile Applications
 - Characteristics
 - Benefits
 - Overview of Available Technologies
- 1.2 Mobile Application Design
 - Application Model and Infrastructure
 - Hardware and Software Architecture
 - Managing Resources
 - Development Workflow

2 Development of Mobile Applications with MIDP and Symbian

- 2.1 Mobile Graphics
 - User Interface
 - Scalable Vector Graphics
 - Mobile 3D Graphics (M3G) API
- 2.2 Security
 - Secure Design
 - MIDP security features
 - Security for Symbian OS
- 2.3 Networking
 - Basic Connectivity
 - Bluetooth Connectivity
 - Web Services

3 *Technology Integration*

Design, develop and deploy a commercial application for Mobile Phones.

Practical Programme:

This module has a strong emphasis on practical applications. Lectures will be used to introduce new topics and their related concepts. Computer based practicals will be used to illustrate principles introduced in the lectures.

Best practice in Object Oriented and Mobile Development will be introduced and tested in the submitted assignment.

Students will be required to submit one completed assignment. This assignment is to be presented by the students late in the module and expects students to develop a commercial Mobile Application. This assignment should show a good understanding of the design process and implementation of Mobile Application.

Media Processing

Learning Outcomes:

On successful completion of this module, the student should be able to:

1. Generalise competency in compression theory
2. Apply compression to a still images, video and audio data to produce a variety of container formats
3. Demonstrate an understanding of multimedia synchronisation by developing a piece of audio-visual material.
4. Understand the concepts and issues surrounding delivery of multimedia content over networks
5. Be able to develop dynamic multimedia software using the Java Media Framework (JMF)

Syllabus Content:

The aim of the module is to give the student a deeper understanding of multimedia content by studying compression, format encoding and synchronisation. The concepts surrounding compression theory will be introduced and then expanded through study of some of the common multimedia formats, such as MPEG2, MPEG4, AC3, AAC, JPEG and MP3. The students will use image, video and audio editing tools to gain experience with these formats.

The module will introduce the student to the Java Media Framework (JMF API) and its associated player, which are a set of software development tools for the Java platform that allow the dynamic capture, playback, streaming and transcoding of multiple media formats.

Particular emphasis will be placed on issues surrounding the delivery of multimedia content in a networked environment, including performance, quality, streaming and dynamic encoding.

Continuous assessment will be based on the student delivering professional-grade networked multimedia systems, including dynamic on-the-fly conversion.

Pre-requisites:

1. Basic understanding of computer architecture and of operating systems.
2. Programming experience in an Object Oriented Programming Language (graduate of a Honours BSc or BEng involving significant programming experience).

Indicative syllabus content:

1. Compression

- Algorithmic complexity theory
- Information entropy
- Lossy vs. Lossless compression
- Psychoacoustics

2. Multimedia Encoding Standards

- Still Image (JPEG)
- Video coding standards (MPEG, H.263)
- Audio Coding Standards (MP3, AC3, AAC)

3. Synchronisation

- Temporal relationships
- Merging of Audio, Still Image and Video

4. Multimedia Development

- Java Media Framework
- Design and Develop professional level multimedia systems

4. Networked Multimedia

- Communication and Networking Issues
- Quality and Performance

- *Dynamic Encoding*

Practical Programme:

This module will be a blend of theory-based lectures and practical laboratories. The focus is on the utility of the concepts discussed in lectures and their application in modern multimedia software development. Practical activities will be very hands-on, with students encouraged to adopt agile methods of development.

Assessment shall consist of weekly lab work and two large assignments. The first assignment will ask the student to demonstrate familiarity with the theoretical concepts covered in lectures by creating a piece of synchronised multimedia that contains video, still images and audio. The student will deliver this project in a number of formats reflecting a variety of compression criteria. The second assignment will be the development of a dynamic web-based multimedia delivery application using the Java Media Framework.