

<p>4 days</p> <p>021</p>	<h2 style="text-align: center;">UML for Software Development</h2> <p><i>The Unified Modelling Language has become one of the industry standard notations for the analysis and design of IT systems. This course introduces the UML 2.0 and its application to modern software development processes. Participants receive a solid grounding in OO technology and are then taken through the software development cycle using the UML and addressing essential elements such as Testing and Database design. There are extensive exercises and opportunities for discussion. This is an intensive course to show how to design robust OO systems. The use of the UML in forming a basis for Requirements Gathering through to Testing, Acceptance and Delivery is emphasized, enabling participants to deliver high-quality surprise-free systems. All exercises come with worked solutions and delegates receive a copy of “UML Distilled” by Martin Fowler</i></p>	
<p><b>Course Objectives</b></p> <ul style="list-style-type: none"> <li>▪ Explain Object Oriented design and development techniques and terminology</li> <li>▪ Show how OO Analysis and Design relates to traditional techniques</li> <li>▪ Describe how the use of the UML for modelling fits with OO technology for software development</li> <li>▪ Show how the use of Abstraction, Inheritance and Polymorphism can simplify programs.</li> <li>▪ Describe Requirements capture and their management with Use Cases</li> <li>▪ Show how static and dynamic attributes can be modelled in the UML</li> <li>▪ Explain why Dynamic modelling leads to more complete solutions</li> <li>▪ Show how to design successful normalised and optimise de-normalised relational databases.</li> </ul>	<ul style="list-style-type: none"> <li>▪ Show how UML is used to model business processes and avoid inconsistent design.</li> <li>▪ Show how complex business rules can be captured in Decision Tables and how these are introduced into Use Cases</li> <li>▪ Describe the phases and workflows of the unified process and how they are applied with UML</li> <li>▪ Show how UML is used to manage a smooth transition from Analysis to Design and Implementation</li> <li>▪ Present the Package and Architecture modelling features of UML</li> <li>▪ Present Patterns as a way of simplifying programs and maximising reuse.</li> <li>▪ Present MDA and how it can be used to reduce development effort.</li> <li>▪ Show how Testing is implemented throughout a UML-based project</li> </ul>	<p><b>Audience</b></p> <ul style="list-style-type: none"> <li>▪ Management wanting to understand the project issues of UML</li> <li>▪ Technicians wanting to understand analysis and design in a UML world</li> <li>▪ Senior staff needing to examine the potential of UML for their organization</li> <li>▪ Anyone concerned with Requirements Gathering and Program Design</li> </ul> <p><b>Prerequisites</b></p> <ul style="list-style-type: none"> <li>▪ General knowledge of software development process.</li> <li>▪ Keen to provide more professional IT services</li> </ul> <p><b>Timetable</b>  Register at 09:00 for 09:30 start on Day1, 09:00 the rest of the course.  Finish at 17:00 on all days.</p> <p><b>Presentation Style</b>  Lectures, demonstrations, group discussions and exercises</p> <p><b>Dates and Venues</b>  Refer to <i>Course Schedules</i>.</p>

OO61

4 days

# Systems Analysis with the UML

The course covers:

- 📁 Introduction to Software Development Processes
- 📁 Process Alternatives
- 📁 The Rational Unified Process
- 📁 Faults in our Processes
- 📁 Iteration as a General Technique
- 📁 Waterfall, XP, SCRUM and Spiral
  
- 📁 Object Orientation in Business
- 📁 Benefits of Object Technology
- 📁 Objects and Classes
- 📁 Objects and States
- 📁 Abstraction
- 📁 Concrete and Abstract design
- 📁 Inheritance Pitfalls
- 📁 Polymorphism and inheritance
- 📁 CRC Techniques
- 📁 Interface versus implementation
  
- 📁 UML Overview
- 📁 Origins and purpose of UML
- 📁 Overview of the UML diagrams
- 📁 Using UML diagrams
- 📁 What's new in UML 2.0 and why
- 📁 The UML Meta Language
- 📁 The OCL
  
- 📁 Decision Tables
- 📁 Business Rule Capture and implementation
- 📁 Completeness Checking
- 📁 Business Terminology
- 📁 Code Generation from Tables
- 📁 Decision Tables in Use Cases
- 📁 Test Case creation

- 📁 Use Cases
- 📁 What are Use Cases
- 📁 Constructing Use Case diagrams
- 📁 Requirements and Use Cases
- 📁 Documenting Use Cases
- 📁 Alternates and Exceptions
- 📁 Scenarios
- 📁 Business Rules with Decision Tables
- 📁 Test Case creation
  
- 📁 Static Modelling
- 📁 Classes and interfaces
- 📁 Class diagrams
- 📁 Object diagrams
- 📁 Class relationships
- 📁 Robustness Analysis and Class diagrams
- 📁 Association and Association classes
- 📁 Aggregation and Composition
- 📁 Inheritance
- 📁 Abstract classes
- 📁 Interfaces
- 📁 Stereotypes
- 📁 Constraints
- 📁 Use Extensions to UML
  
- 📁 Dynamic Modelling
- 📁 Introduction to UML dynamic modelling
- 📁 Workflow and Activity diagrams
- 📁 Partitioning with Swimlanes
- 📁 Sequence diagrams
- 📁 Collaboration diagrams
- 📁 Events, Signals and Guards
- 📁 Statecharts and Statetables
  
- 📁 Objects and Databases
- 📁 Object Persistence
- 📁 Mapping Classes to Relational Tables
- 📁 Factory Classes and Proxies
- 📁 OO Databases
- 📁 Relational Databases
- 📁 Object-Relational
- 📁 JDO, Hibernate and related mapping tools

- 📁 Component and Deployment Models
- 📁 Using Packages to organise your system
- 📁 Using components to model physical organisation
- 📁 Modelling hardware architecture
  
- 📁 What are Patterns?
- 📁 Designing with Patterns
- 📁 Where to find Patterns
- 📁 Frameworks
- 📁 Some Core Patterns
- 📁 Delegation
- 📁 Factory
- 📁 Singleton
- 📁 Decorator
- 📁 Pros and Cons of Patterns
  
- 📁 Model-Driven Architecture
- 📁 The MOF
- 📁 The CWM
- 📁 The PIM and PSM
- 📁 Transformations
- 📁 PIM to PSM examples
  
- 📁 Testing
- 📁 From Use Cases to Acceptance tests
- 📁 Class-oriented testing
- 📁 Test Frameworks
- 📁 Test-driven development
- 📁 Test metrics
  
- 📁 Summary and Conclusion
- 📁 A strategy for Beginning
- 📁 Where to go for Information
- 📁 Avoiding mistakes
- 📁 Commonsense approach to introducing the UML