

## ***The OOAD using UML Program***

---

This UML training course provides an in-depth technical study of object oriented analysis and design for computer systems. Modeling techniques are based on the Unified Modeling Language with special emphasis on specifying requirements with use cases and how to convert these into a related design. Each technique is taught to the level required for competence on a real project and understanding is tested and improved with case study. A short, smooth, iterative and incremental software process is described which improves speed of delivery, estimation and traceability. Students spend a significant part of the training course applying the techniques to a real-world project example either on paper or using a suitable case tool.

### **Duration: 4 Days**

### **Program Objectives**

- ✓ Become familiar with all phases of Object-Oriented Analysis and Design (OOAD)
- ✓ Master the main features of the Unified Modeling Language (UML)
- ✓ Master the main concepts of Object Technologies and how to apply them at work
- ✓ Develop the ability to analyze and solve challenging Problem Domains
- ✓ Learn the Object Design Principles and understand how to apply them towards implementation

### **Suitable for:**

*Project managers, system analysts, system designers, developers, programmers, anyone who is moving to, or using object technology and wishes to use the Unified Modeling Language for analysis and design.*

### **Pre-requisites:**

- Some experience of software engineering
- Some prior understanding of objects would be useful, but is not essential

## **Day 1**

### **❖ The Genesis of UML**

- ❑ Analysis and Design Methods
  - What is the Purpose of a Method?
  - From Functional to Object-Oriented Methods
  - The Proliferation of Object-Oriented Methods
  - Booch and OMT Getting Closer
- ❑ Unification of Methods
  - Towards a Unified Modeling Language
  - Model and Metamodel

### **❖ The Object-Oriented Approach**

- ❑ Need For Object Orientation
- ❑ The basic Object Oriented Approach to Software development
- ❑ Overview of Object Oriented Analysis
- ❑ Overview of Object Oriented Design

### **❖ The UML Notation**

- ❑ Introduction to Visual Modeling
- ❑ Evolution of Visual Modeling Technique
- ❑ UML Notation
- ❑ Basic Concepts
  - Common Elements
  - Common Mechanisms
    - Stereotypes
    - Tagged Values
    - Notes
    - Constrains
    - Dependencies
    - Type/Instance and Type/Class Dichotomies
  - Data Types
  - Packages

### **Basic Modeling workshop**

## **Day 2**

### **❖ The Use Case Analysis**

- ❑ The Requirement Analysis from a Problem statement
- ❑ Deriving the analysis process from the problem statement
- ❑ Identifying Use cases, Business Sequences and Analysis Classes for the business model
  
- ❑ Use Case Diagrams

- The Use Case Model
- Relationships Between Use Cases
  - The Communicates Relationship
  - The Uses Relationship
  - The Extends Relationship
  
- Converting the Business model in to the Functional Model
- Deriving the Logical Sequence / Collaboration Diagrams Deriving the Functional Class
- Deriving the Class Relationships
  - Associations
  - Aggregations
  - Composition
  - Generalization

### *System Use Case Workshop*

## **Day 3**

### **❖ The Object Behaviour**

- Representing the State transition for a Concrete / Action Object
- Statechart Diagrams
  - State Machines
  - States
  - Transitions
    - Events
    - Guards
    - Operations, Actions and Activities
    - Execution Points of Operations
- Representing the State transition for more than one related Concrete / Action Objects
- Activity Diagrams
  - Representation of Activities

### *Object Relationship Workshop*

## **Day 4**

### **❖ The Architectural Diagrams**

- Representation of the Functional Architecture interacting with the Non-functional Architecture using component Diagrams
- Component Diagrams
  - Components
  - Dependencies Between Components
    - Subsystems
    - Processes
    - Integration with Development Environments

- Representation the Non-Functional requirements of the systems in a Deployment Diagram
  - Deployment Diagrams
  - Representation of Nodes

\* \* \* \* \*