# Unified Modeling Language

## UML Views of a Project, Diagramming Notation, Use Cases, Classes, Relationships, Behaviors, States, OCL

UML is used by object-oriented designers to unambiguously specify an OO design, to discuss it with other designers and to communicate it to all stakeholders - developers, end-users and management.

We can think of UML as a graphical design language for object oriented software. It has become the graphical lingua franca of object design, supported by most design/development tools and understood by most software professionals. To discuss modern software among a team of stakeholders, we need to describe objects – their internal static information, how they relate to each other, how they behave dynamically (both intra- and inter-object) and how they are delivered to end-users. Software systems are becoming increasingly complex and at the same time there is persistent pressure to decrease development costs and time scales, and improve quality.

UML is the key to succinctly describing correct software architecture, and this in turn is the cornerstone of successful projects. A very important point is such design needs to be sufficient without being excessive.

### Target Audience

Experienced object-oriented architects and developers who need a detailed understanding of all aspects of UML.

### Prerequisites

A good understanding of object-oriented principles and previous OO design & programming experience.

### *Contents of One-Day Training Course*

#### UML Introduction

"The UML is a language for visualizing, specifying, constructing and documenting the artifacts of a software-intensive system" (Booch et al)

#### UML Views

There are five ways of looking at systems: User, Design, Process, Implementation and Deployment views
UML diagramming notation
Class name, operation and attribute, interface, component, package, note, state, state transition, event, action, etc.

#### UML Diagram Types

Use cases, class, object (instance), sequence, collaboration, statechart, activity, component, deployment

#### Classes and Relationships

The structural (static) aspects of a project
Completely defining operations/attributes
Generalization

#### Advanced Classes and Relationships

The four ways classes relate to others
Aggregation / composition

#### Interfaces/Components/Deployment

Functionality as components
Collaborations

#### Dynamic Behavior

The dynamic aspects of a project
Interaction among a set of object instances
Finite state machine within an object (statechart diagrams)
General system activity (activity diagrams)

#### UML For Database Design

The Persistent tagged value
Storing objects in a relational database
Generating database schemas from UML

#### Object Constraint Language

OCL enables tighter specification by adding detailed constraints to elements within object models

#### Round-Trip Engineering

Converting between UML and languages such as C++, Java, C#

#### UML and ...

Multithreading
Networking
User Interface
Frameworks

#### UML and Agile Development

Requirements, Analysis, Design, Implementation, Test, Delivery – all progressing in parallel

#### Design: sufficient, not excessive

How much/little design does an agile actually project need?
How to go about creating it?
UML as a form of sketching for devs

#### Complete Example

Walk through of creating a set of UML diagrams for the lifecycle design of a modern multithreaded HTTP web server