

Hands-on course , 4  
day(s)  
Ref : UML

## Participants

This course is aimed at project leaders, analysts, designers, software architects and developers involved in developing object-oriented projects.

## Pre-requisites

Participants must have a general knowledge of information technology. Experience in analysis and design, and knowledge of an object-oriented language, are not required but would be an advantage.

## Next sessions

# UML 2.0, Enterprise Modeling

## OBJECTIVES

*UML (Unified Modelling Language) is the standard notation that has become indispensable for modelling information systems. It makes it possible to specify, view, construct and document all the artefacts of the system, and is suited just as well to information systems as it is to software, technical, business or real-time systems. This course will allow you to acquire the knowledge required to use UML and to implement the best practices of object analysis and design. Examples and actual case studies reveal UML's technical details and the different steps in constructing an IT solution. At the end of this course, the participants will be able to evaluate the benefits obtained from using UML and object-oriented design in project development.*

### 1) The Object approach

### 2) Object analysis and design, an introduction to UML

### 3) UML diagram of cases of use

### 4) UML diagrams of static modelling

### 5) UML diagrams of dynamic modelling

### 6) Finalising the system

### 7) Advanced concepts, tools

## Workshop

*A range of case studies makes it possible to learn the various phases of object modelling in UML. Environments, drawn from industry or the Open Source community, are presented for demonstration purposes. The exercises and case studies are performed using Rational Rose or XDE tools.*

## 1) The Object approach

### Understanding the main notions of the object approach

- Review of the programming paradigms (logic, imperative, object, etc).
- Objects: identity, state and behaviour. Relationship to the real world and information systems.
- The popularity and advantages of the Object-Oriented approach.
- Abstraction, encapsulation, classification. Classes and instances. Abstract classes. The concept of inheritance.
- Methods and sending messages between objects. Polymorphism. Overloading and re-definition.

## 2) Object analysis and design, an introduction to UML

### Why model? Learning about the analysis and design spectrum

- The application area and modelling an IT solution. The model, a central artefact of the project procedure.
- Analysing and designing an IT solution. The impacts of programming languages.
- Changing over to Object analysis/design. Advantages.

### General presentation of UML

- The background, changes and objectives. Architect's views.
- The core of UML: the different types of diagrams. The differences between static and dynamic diagrams.
- Presentation of several modelling approaches.
- UML extensions: stereotype, profiles, constraints, etc.

## 3) UML diagram of cases of use

### Capturing and describing the application's functional needs

- The main objectives and use. Describing the functionalities of the system.
- The elements of the diagram: use cases, players and the system boundary.
- Steps in constructing the use case model.
- How do you identify the players? How do you describe a use case? The scenarios.
- Formats, pre-conditions, post-conditions, relationships (use, inclusion, extension).

## 4) UML diagrams of static modelling

### Showing a view of the whole system, these elements and their relationships

- Class diagram: its role and its use. Examples.
- How do you identify useful classes?
- A class in UML: name, attributes and operations. Visibility (public, private, and protected).
- Formalism and notation.
- Relationships between classes (association, generalisation, aggregation and composition).
- Multiplicities, roles, constraints, etc.
- Abstract classes, interfaces, packages.
- Diagram of class instances and objects.

## 5) UML diagrams of dynamic modelling

### Showing how the system changes and the interactions between objects

- Sequence diagram: interactions between objects over time. Message (synchronous and asynchronous).

- Collaboration diagram: role of the objects, interactions, competing processes, etc
- Transition state diagram: possible states of an object and events triggering the transitions.
- Activity diagram: flow of activities to carry out an operation, the objects in charge of these activities.
- Notations and examples.

## 6) Finalising the system

### The system's software and hardware architecture

- Architecture models. Organisation in layers. Sub-systems.
- Packages and their relationships.
- Component diagram: organising the code in modules, dependencies.
- Deployment diagram: the physical deployment of the system (machines, networks, etc).

## 7) Advanced concepts, tools

### Design supplements

- Data models. Object/Relational Mapping. Other models.
- Designing HMI screens.
- Responding to recurrent problems
- Design Patterns (e.g.: singleton, adapter, proxy, facade, etc), their role in design.
- Frameworks, re-use.

### Modelling working groups

- Generating reports, code. Creating stereotypes, etc.
- XML format for exchanging UML models between AGLs.

### UML and project methodology

- Various approaches. Integrating UML.
- Introduction to Rational Unified Process (RUP), iterations, phases and activities.
- Other possibilities (XP, etc.).