

# Object-Oriented Analysis and Design with UML

<b>Course No.</b>	1516
<b>Description</b>	This course provides a hands-on introduction to object oriented analysis and design with UML. Participants are introduced to several tools for analysis and design including; Use Case narratives and diagrams, Class diagrams, Sequence and Collaboration diagrams, State and Activity diagrams and design pattern principles. Practice using these tools is accomplished through extensive labs in a team environment taking an application through all phases of analysis and design. This course also covers fundamental object oriented concepts such as polymorphism, inheritance, encapsulation and interfaces. Complete solutions are provided using UML notation.
<b>Audience</b>	This course is recommended for Programmers, IS business analysts and managers who will be participating in object analysis, design and implementation.
<b>Objectives</b>	<ul style="list-style-type: none"><li>• Understand the major concepts of object orientation</li><li>• Understand the activities and deliverables in each phase of an object oriented project using the Unified Process</li><li>• Avoid the pitfalls common to initial object-oriented projects</li><li>• Identify classes and responsibilities from Use Cases</li><li>• Create class diagrams with UML notation</li><li>• Create dynamic models with UML notation (sequence, collaboration, state and activity diagrams)</li><li>• Consider coupling and cohesion issues in designing an object system</li><li>• Understand why composition provides more flexibility than inheritance</li><li>• Discuss designing with interfaces</li></ul>
<b>Major Topics</b>	<ul style="list-style-type: none"><li>• Object-Oriented Fundamentals</li><li>• Object-Oriented Analysis</li><li>• Object-Oriented Design</li></ul>
<b>Duration</b>	4 days
<b>PDU's</b>	28



## Course Contents

### 1. Object-Orientation - A Conceptual Look

- How we think
- The Functional Approach
- Changing Requirements
- An Example
- Object Orientation
- How It Works

### 2. Object-Orientation – Basic Principles

- Objects
- Attributes
- Behavior
- Identity
- Classes
- Instantiation
- Inheritance
- Polymorphism
- Abstract Classes
- Visibility
- Encapsulation

### 3. UML and Unified Process Overview

- UML History
- UML Diagrams Elements
- Architecture
- Unified Process
- Use Case Driven
- Architecture Centric
- Iterations and Increments
- Major Phases

### 4. Use Cases

- Explained
- Example
- Advantages
- Basic Notation
- Role in Requirements

### 5. Use Case Specification

- Identifying Actors
- Actor Roles
- Goal levels
- User Goals
- Goal Example
- Summary Level Goals
- Subfunction Goals
- Work Summary

### 6. Detailing a Use Case

- Use Case Structure
- Writing Guidelines
- Extensions

### 7. Use Cases – Odds and Ends

- Use Case Relationships
- Include
- Extend
- Generalization
- Technology Variations
- CRUD scenarios
- When are you finished

### 8. Activity Diagrams

- Action States
- Transitions
- Branches
- Forks and Joins
- Swimlanes
- Object Flows

### 9. Finding Classes

- Analysis Workflow
- Analyze a Use Case
- Analysis Classes
- Analysis Class Notation
- Boundary Classes
- Entity Classes
- Control Classes
- Notation
- CRC Analysis



## 10. Object Interactions

- Realizing Use Cases
- Interaction Diagrams
- Collaboration Diagram Example
- Collaboration Diagram Notation
- Messages
- Message Stereotypes
- Multiobjects
- The Process – First Pass
- Assigning Responsibilities
- Expert, Coupling, Cohesion, Creator
- Example Revisted
- Sequence Diagram Example
- Sequence Diagram Notation
- Message Types

## 11. Class Diagrams

- Analysis Class Diagrams
- Associations
- Multiplicity
- Reflexive Associations
- Navigability
- Associations and Attributes
- Association Classes
- Dependencies
- Generalization
- Bad Generalization
- Diagram Example

## 12. Packages

- Packages Described
- Package Notation
- Dependencies
- Transitivity
- Cyclical Dependencies
- Example
- Architectural Layers

## 13. Design

- Design Workflow
- Design vs. Analysis Model
- Architectural Design
- Software Layers
- Design a Class
- Attributes and Methods

- Design Class Notation
- Design Class Principles
- Complete, Primitive, High Cohesion, Low Coupling
- Design Relationships
- Aggregation and Composition
- Identifying Whole/Part
- Collections
- Design a Use Case
- J2EE Example
- Components
- Deployment Diagrams

## 14. Designing with Inheritance and Composition

- Composition vs. Inheritance
- Composition Example
- Inheritance Types
- Benefits of Inheritance
- Risks – Weak Encapsulation
- Changing Subclasses
- When to Inherit
- Passing the Test
- Using Composition
- Combining the Two - Strategy
- Validating the New Model

## 15. Interfaces

- Explained
- Notation
- Interface Polymorphism
- Interfaces in Actions
- Finding Interfaces
- Subsystems and Interfaces
- Facades
- Using Interfaces

## 16. State Transitions

- Statecharts
- States
- Basic State Machines
- State Syntax
- Transitions
- Timed Transitions
- Substates
- History
- Concurrent States

