

## UML 2: A Comprehensive Hands-On Introduction - 3 Days

### *Course 216 Overview*

- You Will Learn How To**
- Model software and nonsoftware systems using UML 2
  - Capture and document user requirements using use cases
  - Generate and interpret UML models using the complete diagramming notation
  - Use CASE tools to build and manipulate fully featured UML models
  - Ensure consistency and accuracy throughout all diagrams
  - Represent design patterns in UML
- Course Benefits** The Unified Modeling Language (UML) is the industry-standard notation for producing the models of a system. In this course, you learn to generate and interpret UML models as applied to a wide range of activities using the significant extensions and enhancements of UML 2. These skills are put into practice using a market-leading CASE tool.
- Who Should Attend** Designers, programmers, project managers and all other personnel involved in systems development. UML practitioners who wish to update their skills to UML 2 will also benefit. Knowledge of object-oriented techniques is helpful but not required.
- Hands-On Training** You gain hands-on UML experience using CASE tools. Exercises include:
- Modeling system requirements and business processes with use cases
  - Forward- and reverse-engineering between UML models and code
  - Representing system structure using class and object diagrams
  - Modeling behavior with interaction, state machine and activity diagrams
  - Generating HTML and textual documentation
  - Producing interrelated diagrams in a large system model

## UML 2: A Comprehensive Hands-On Introduction - 3 Days

### Course 216 Outline

#### Introduction to UML

##### Speaking a common language

- The importance of modeling
- Enabling concise communication
- The evolution of UML

#### Elements of UML

- Building blocks: things, relationships and diagrams
- Architectural views: use case, design, implementation, process and deployment
- Levels of detail: visualization, specification and construction

#### Object-oriented concepts

- Objects and classes
- Links and relationships
- Inheritance and polymorphism

#### Modeling the Structure of a System

##### Specifying classes

- Modeling user-defined types as classes
- Representing information as attributes
- Representing functionality as operations

##### Identifying relationships between classes

- Dependencies
- Associations
- Aggregation and composition
- Generalization

#### Object and class diagrams: the core of UML

- Showing classes and their relationships
- Depicting snapshots using object diagrams
- Defining information models with class diagrams

#### Modeling the Behavior of a System

##### Use case diagrams: describing user requirements

- Representing systems boundaries
- Actors and use cases
- Notations for refinement

##### Sequence and communication diagrams: depicting typical event scenarios

- Events and signals
- Showing time-ordered behavior
- Expanding use cases into the developers' view
- Converting between sequence and communication diagrams

#### Expressing real-time aspects

- Synchronous/asynchronous messages
- Representing timing constraints and transmission delays
- Implementing timing diagrams

#### Representing State Machines

##### State machine diagrams: capturing state-dependent behavior

- States, transitions and events
- Concurrent substates
- History and synch states

##### Activity diagrams: specifying behavioral logic

- Modeling workflows
- Partitioning activities using swimlanes
- Concurrency and synchronization of parallel activities

#### Architectural Modeling

##### Packages and interfaces

- Distinguishing between classes/interfaces
- Exposing class and package interfaces
- Subscribing to interfaces

#### Component and deployment diagrams

- Describing dependencies
- Deploying components across threads, processes and processors
- Describing internal structure using composite structure diagrams

#### Design patterns

- Patterns, mechanisms and frameworks
- Representing design patterns
- Referencing design patterns

#### Applying UML

##### Model-Driven Architecture (MDA)

- The Meta-Object Facility (MOF)
- Common Warehouse Meta-model (CWM)

#### Life cycle stages

- Using UML within the Unified Process
- Modeling business processes
- Capturing requirements
- Systems analysis
- Software design