Object Oriented Analysis & Design with UML

Duration: 4 days *(Face-to-Face & Remote-Live)*, or 28 Hours *(On-Demand)*

Price: CDN$2,775 *(Face-to-Face & Remote-Live)*, or CDN$1,995 *(On-Demand)*

Discounts: We offer multiple discount options. Click here for more info.

Delivery Options: Attend face-to-face in the classroom, remote-live or on-demand training.

Students Will Learn

- Extracting a system's requirements using a use-case driven approach
- Leveraging the experience of experts by applying analysis and design patterns
- Defining a set of extensible, reusable software classes (a class library) for the problem domain
- Building interaction diagrams that define the interactions among the objects that are required to achieve the desired system behavior
- Defining a set of candidate classes that suitably model a problem domain
- Establishing metrics, peer reviews and heuristics to improve the quality of the object models
- Effectively documenting all phases of the software process using UML
- Applying an iterative and incremental approach to construction of software systems and components

Course Description

This OOA&D training course presents the key concepts and methodologies required to perform quality object-oriented software engineering, with particular attention to practical techniques such as use-case and CRC analysis, UML diagramming, and patterns. Students practice applying object oriented analysis during the course to improve software designs and to see how software objects can be altered to build software systems that are more robust and less expensive. Students use several methods for analyzing software systems, finding and refining useful classes and relationships between objects. Care is taken not to focus on any one language so that all students can participate in the design exercises without relying on specific programming skills. The course emphasizes the most practical analysis and design methods, including the application of use case analysis, CRC analysis, problem domain analysis, activity diagramming, interaction diagramming, and class diagramming. The Unified Modeling Language (UML) is presented in detail and is used in the exercises and case studies. Practical aspects of project management and implementation are presented from the perspective of experienced object system designers. Special emphasis is given to the use of object patterns in developing software systems. The students apply their skills in labs that are mini design sessions, during which the instructor helps the students identify and overcome common obstacles that occur

during group sessions.

Course Prerequisites

Knowledge of structured programming concepts.

Course Overview

The Object Paradigm

- Objects and Classes
- Abstraction and Encapsulation
- Methods and Messages
- Interfaces, Inheritance, and Polymorphism
- Access Control
- The Business Case for OO Development

Managing and Participating in the OOA&D Approach

- Information Gathering Techniques
- Group Orientated Problem Solving
- Brainstorming, Role-Playing
- Managing Complexity via the "Iterative and Incremental" Approach
- Managing Design Sessions
- Design vs. Implementation
- Quick Prototyping
- Validation and Quality

Diagramming & Notational Techniques Using the UML

- Overview of Analysis and Design Phases
- UML Notation
- Analysis Diagramming Techniques
- Design Diagramming Techniques
- Generalization/Specialization
- Aggregation and Composition
- Association, Cardinality, Navigability
- Package and Deployment Diagrams
- Icons, Relationships, and Adornments

Requirements and Analysis Phase

- System Functions, Features and Constraints
- Behavioral Analysis
- Domain Analysis
- Identifying Use Cases
- Use Case Descriptions
- Using CRC Cards
- Containment and Composition
- Referential Aggregation
- Inheritance, SubTypes and Is-A Hierarchies
- Association and Link Relationships
- Diagramming System Events
- State Transition Diagramming

Design Phase

- Translating Analysis Concepts into Software Classes
- Optimizing Classes and Objects: The Multi-Tiered Architecture View
- Mapping System Functions to Objects
- Object to Object Visibility
- Collaboration Diagrams
- Sequence Diagrams
- Specifying Object Interfaces
- Specification Class Diagrams

Patterns

- Benefits of Patterns
- Using Patterns During Analysis
- Using Patterns During Design
- Design Patterns (Gang-of-Four Format)
- GRASP Patterns
- Model-View-Controller Pattern
- Persistence Patterns
- Patterns as Internal Documentation

Design Refinement

- Designing for Extensibility

Project Management and Implementation Issues

- Designing for Reusability
- Partitioning the Class Space
- Checking Completeness and Correctness
- Testing Business Processes
- Design Metrics
- Discovering Reusable Patterns
- Planning for Reusability
- Transition Strategies and Planning Legacy System Integration
- Managing the Development Cycle
- Partitioning Work
- Source Code Organization
- Choosing Tools and Languages
- Software Quality Metrics

OO Languages and Tools

- Survey of OO Languages
- The Role of Class Libraries
- The Role of OOA&D Tools

Advanced Design Concepts

- Expanding Inheritance Hierarchies
- Abstract Classes and Virtual Methods
- Overriding and Overloading
- Multiple Inheritance
- Interface versus Implementation Inheritance

Persistent Object and Database Issues

- The Coad Data Management Domain
- Object Persistence
- Object-Orientated Database Management Systems (ODBMS)
- Object Orientated versus Relational Databases
- Mapping Objects to Relational Data Structures

Hands On Technology Transfer
The Best Way to Transfer Technology Skills

1 Village Square, Suite 8
14 Fletcher Street
Chelmsford, MA 01824