

Object-Oriented Analysis and Design Using UML

Course: 611 Duration: 4 days

You Will Learn How To

- Capture user requirements in use cases and transform them into detailed designs
- Exploit the rich object-oriented modeling provided by Unified Modeling Language (UML)
- Adapt to changing requirements with iterative techniques and component-based design
- Design solutions optimized for modern object-oriented languages and platforms
- Apply proven design patterns to refine analysis and design models

Course Benefits

Object-Oriented Analysis and Design (OOAD) is the principal industry-proven method for developing reliable, modular, testable programs and systems. This OOAD training course provides practical skills in the latest OO requirements gathering, analysis, design, and testing methods. Intensive hands-on exercises offer you a working knowledge that turns concepts into practice. Students will not just learn UML diagrams. They will learn how to apply UML in the context of OO software development. From requirements gathering to actually mapping the diagrams to code, concepts in this course are consistently illustrated using a single case study - a POS system. This course emphasizes on applying different design patterns to solve common OO design issues. Knowing how to create properly

designed object-oriented systems is what separates programmers from software architects.

Who Should Attend

Anyone involved in developing systems on modern object-oriented platforms. Project teams benefit greatly by sharing the same methodology with co-developers or with supportive management. Familiarity with an object-oriented language is assumed.

Course Content

Introduction

- Object-Oriented Analysis and Design
- Iterative, Evolutionary, and Agile Development
- Introduction to the Case Study

Object-Oriented Analysis and Design

- Definition: Object-Oriented Analysis and Design (OOA/D)
- Quick OOA/D Example
- Overview of UML

Iterative Development

- Definition: Iterative Development
- Iterative vs Waterfall Model
- Phases on the Unified Process
 - Inception
 - Elaboration
 - Construction

Object-Oriented Analysis and Design Using UML

Course: 611 Duration: 4 days

- Transition

Use Cases

- Requirements
- Use Cases
- Actors
- Finding Use Cases
- Naming Use Cases
- Writing Use Case Text
- Drawing Use Case Diagrams
- Relating Use Cases includes and extends

Domain Models

- Introducing the Case Study NextGen POS System
- The Elaboration Phase
- Definition: Domain Model
- Finding Conceptual Classes
- Classifying Attributes vs Classes
- Discovering Description Classes
- Adding Associations
- Applying UML Notation for Associations
- Adding Attributes
- Applying UML Notation for Attributes
- Guidelines:
 - Defining New Data Type Classes
 - Relating Classes with Associations vs Attributes
 - No Attributes Representing Foreign Keys
- Applying Domain Modeling in NextGen POS

System Sequence Diagrams

- Definition: System Sequence Diagram
- SSD's relationship with other artifacts
- Creating the SSD
- Naming System Events and Operations
- Applying SSDs in NextGen POS

Operation Contracts

- Definition: Operation Contracts
- Operation Contracts' relationship with other artifacts
- Sections of a Contract
- Creating Contracts for System Operations
- Writing Post Conditions
- Looking For State Changes
 - Instance Creation and Deletion
 - Attribute Modification
 - Associations Formed and Broken
- Applying Operation Contracts in NextGen POS

Logical Architecture

- Definition: Logical Architecture
- Applying UML Notation to Logical Architecture
 - Showing Package Dependency
 - Showing Nested Packages
- Designing with Layers
 - Avoid Lower-to-Higher Layer Coupling
 - The Model-View Separation Principle

Object-Oriented Analysis and Design Using UML

Course: 611 Duration: 4 days

- Mapping UML Packages and Layers to Code
- Layers' relationship with other artifacts
- The Facade Design Pattern

Object Design

- Dynamic vs Static UML Models

Interaction Diagrams

- Sequence Diagrams
 - Lifeline Boxes and Lifelines
 - Messages
 - Execution Specification Bar
 - Reply or Returns
 - Calls to "self" or "this"
 - Instance Creation
 - Object Destruction
 - Diagram Frames and Frame Operators
 - Looping
 - Conditional Messages
 - Mutually Exclusive Conditional Messages
 - Iteration over a Collection
 - Nested Frames
 - Showing Related Interaction Diagrams
 - Invoking Static Methods
 - Asynchronous vs. Synchronous Calls

- Communication Diagrams
 - Links
 - Messages
 - Calls to "self" or "this"
 - Instance Creation
 - Message Number Sequencing
 - Conditional Messages
 - Mutually Exclusive Conditional Messages
 - Looping
 - Iteration over a Collection
 - Invoking Static Methods
 - Asynchronous vs. Synchronous Calls

Class Diagrams

- Design Class Diagrams
- Showing UML Attributes
 - Attribute text notation
 - Association line notation
 - Attribute Visibility
 - The UML Notation for an Association End
 - Showing a Collection Attribute in the UML
- Property Strings
- Note Symbols
- Writing Operations
 - Showing Methods in Class Diagrams
 - The create Operation
 - Operations to Access Attributes

Object-Oriented Analysis and Design Using UML

Course: 611 Duration: 4 days

- Constraints
- Generalization
- Abstract Classes and Abstract Operations
- Final Classes and Operations
- Static Methods and Attributes
- Showing Dependencies
- Interfaces
 - Standard Notation
 - Lollipop Notation
 - Socket Notation
- Aggregation vs. Composition
- Qualified Association
- Association Classes
- User-Defined Compartments
- Active Classes
- Class Diagrams's Relationship with Interaction Diagrams

GRASP: Designing Objects with Responsibilities

- Inputs to object design
- Outputs of object design
- Responsibility-Driven Design (RDD)
- Definition: Design Patterns
- Creator
- Information Expert
- Low Coupling
- Controller
- Applying GRASP Patterns in NextGen POS

Mapping Designs to Code

- Creating Class Definitions from DCDs
- Creating Methods from Interaction Diagrams
- Prioritizing Order of Implementation

More GRASP Patterns

- Iteration-2 Requirements of the NextGen POS System
- Polymorphism
 - Liskov Substitution Principle
 - Polymorphic Messages in Interaction Diagrams
- Pure Fabrication
- Applying GRASP Patterns in NextGen POS

GoF Design Patterns

- Adapter
- Factory
- Singleton
- Strategy
- Applying GoF Patterns in NextGen POS

Activity Diagrams and Modeling

- Definition: Activity Diagrams
- Applying Activity Diagrams Business Process Modeling
- Applying Activity Diagrams Data Flow Modeling
- Rake Symbol
- Decision Symbol

- Signals
- Guidelines in Activity Modeling

- Mapping Objects to the Database Using Object Relational Mapping

State Machine Diagrams

- Definition: State Machine Diagrams
- Events, States, and Transitions
- Transition Actions and Guards
- Nested States
- Applying State Machine Diagrams in NextGen POS

More GoF Design Patterns

- Applying GoF Patterns to Perform Caching
- Applying Proxy Pattern to Perform Failover
- Applying Abstract Factory Pattern to Support Multiple Device Platforms

Domain Model Refinement

- Iteration-3 Requirements of the NextGen POS System
- Use Cases for Iteration-3
- Domain Model for Iteration-3
 - Guideline: Determining when to Generalize
 - Guideline: Determining when to Use Abstract Classes

Deployment Diagrams

- Definition: Deployment Diagrams
- Nodes
- EENs as Property Strings
- Communication Paths
- Artifacts
- Node Instances

More SSDs and Contracts

- Revised SSDs for Iteration-3
- Revised Operation Contracts for Iteration-3

Logical Architecture Refinement

- System Operations and Layers
- Upward Collaboration with Observer Pattern
- Relaxed vs Strict Layer Architecture
- Two-tier vs Three-Tier Architecture

About ActiveLearning, Inc.

ActiveLearning is a leading provider of open-source technology training to IT professionals and managers in the Philippines. The company develops and delivers a broad library of instructor-led courses focused on web development, operating systems, programming languages, databases, computer networks, computer and network security, object-oriented technology, and project management.