



Object-Oriented Analysis & Design (3 Days) ST45010

COURSE GOALS: Analysts, designers & programmers responsible for applying OO techniques in their software engineering projects will gain a better understanding of Object-Oriented Analysis and Design.

PREREQUISITES: Familiarity with structured techniques such as functional decomposition is helpful

LEARNING OBJECTIVES: Upon successful completion of this course the student will have a better understanding of:

- Prerequisites & principles of object thinking
- Object knowledge implicit in eXtreme Programming (XP) & Agile software development
- Object conceptualization & modeling
- Metaphors, vocabulary & design for object development
- Decomposing complex domains in terms of objects
- Identifying object relationships, interactions & constraints
- Relating object behavior to internal structure & implementation design
- Incorporating object thinking into XP & Agile practice

KEY TOPICS:

I. Introduction to Object-Oriented Concepts

- A. Procedural vs OO Programing
- B. Moving from Procedural to Object-Oriented Development
- C. Using UML to Model a Class Diagram
- D. Encapsulation and Data Hiding
- E. Inheritance
- F. Polymorphism
- G. Composition

II. How to Think in Terms of Objects

- A. Knowing the Difference Between the Interface and the Implementation
- B. Using Abstract Thinking When Designing Interfaces
- C. Giving the User the Minimal Interface Possible
- D. Determining Users
- E. Object Behavior

III. Advanced Object-Oriented Concepts

- A. Constructors
- B. Error Handling
- C. The Concept of Scope
- D. Operator Overloading
- E. Multiple Inheritance
- F. Object Operations

IV. The Anatomy of a Class

- A.** The name of the Class
- B.** Comments
- C.** Attributes
- D.** Constructors
- E.** Assessors

V. Class Design Guidelines

- A.** Modeling Real World Systems
- B.** Identifying the Public Interfaces
- C.** Designing Robust Constructors
- D.** Designing Error Handling into a Class
- E.** Documenting a Class and Using Comments
- F.** Designing with Reuse in Mind

VI. Designing with Objects

- A.** Design Guidelines
- B.** Performing the Proper Analysis
- C.** Developing a Statement of Work
- D.** Gathering the Requirements
- E.** Developing a Prototype of the User Interface

VII. Mastering Inheritance and Composition

- A.** Reusing Objects
- B.** Inheritance
- C.** Composition
- D.** Why Encapsulation is Fundamental to OO

VIII. Frameworks and Reuse: Designing with Interfaces and Abstract Classes

- A.** What is a Framework
- B.** What is a Contract
- C.** An E-Business Example
- D.** The UML Object

IX. Building Objects

- A.** Composition Relationships
- B.** Building in Phases
- C.** Types of Composition
- D.** Avoiding Dependencies
- E.** Cardinality
- F.** Tying It All Together

X. Creating Object Models with UML

- A.** What is UML
- B.** The Structure of a Class Diagram
- C.** Attributes and Methods
- D.** Access Destination
- E.** Inheritance
- F.** Interfaces
- G.** Composition
- H.** Cardinality

XI. Objects and Portable Data: XML

- A.** Portable
- B.** The Extensible Markup Language (XML)
- C.** XML vs. HTML
- D.** XML and Object-Oriented Languages
- E.** Sharing Data Between Two Companies

XII. Persistent Objects: Serialization and Relational Database

- A.** Persistent Object Basics
- B.** Saving the Object to a Flat File
- C.** Using XML in the Serialization Process
- D.** Writing a Relational Database

XIII. Objects and the Internet

- A.** Evolution of Distributed Computing
- B.** Object-Based Scripting Languages
- C.** A JavaScript Validation Example

- D. Objects in a Web Page
- E. Distributed Objects and the Enterprise

XIV. Objects and Client/Server Applications

- A. Client/Server Approaches
- B. Propriety Approach
- C. Nonproprietary Approach
- D. Object-Definition Code
- E. Client Code
- F. Server Code

XV. Designing Patterns

- A. Why Design Patterns
- B. Types of Design Patterns
- C. Creational Patterns
- D. Structural Patterns
- E. Behavioral Patterns