

Object-Oriented Analysis and Design (OOA&D) is a process of identifying the needs of a software project and laying out specifications in various models. The Unified Process, in which four iterative phases are used to hone the architecture and build the system, is the framework for many OOA&D endeavors. Each phase can consist of models from different perspectives. These models are documented using the Unified Modeling Language, an industry-standard language for visualizing, specifying, and documenting the architecture of the system.

In this course, students learn how to identify and design objects, classes, and their relationships to each other, which includes links, associations, and inheritance. A strong emphasis is placed on UML diagram notation for use cases, class and object representation, links and associations, and object messages. This course utilizes UML 2.0 notation.

Course Objectives:

- Apply the principals and practices of Object-Oriented Analysis and Design.
- Use modeling in analysis and design, particularly in visual modeling.
- Use the Unified Modeling Language to create visual models of business problems and software solutions.
- Design programs with objects.
- Create more flexible and more maintainable software systems at lower costs.

Audience: Analysts, designers, and programmers responsible for applying OO techniques in their software engineering projects.

Prerequisites: Familiarity with structured techniques such as functional decomposition is helpful.

Number of Days: 5 days

<p>1. Course Introduction Course Objectives Overview Suggested References</p> <p>2. Introduction to Analysis and Design Why is Programming Hard? The Tasks of Software Development Modules Models Modeling Perspective Objects Change New Paradigms</p> <p>3. Objects Encapsulation Abstraction</p>	<p>Objects Classes Responsibilities Attributes Composite Classes Operations and Methods Visibility Inheritance Inheritance Example Protected and Package Visibility Scope Class Scope</p> <p>4. Advanced Objects Constructors & Destructors Instance Creation Abstract Classes Polymorphism</p>
--	---

- Polymorphism Example
- Multiple Inheritance
- Solving Multiple Inheritance Problems
- Interfaces
- Interfaces with Ball and Socket Notation
- Templates
- 5. Classes and Their Relationships**
 - Class Models
 - Associations
 - Multiplicity
 - Qualified Associations
 - Roles
 - Association Classes
 - Composition and Aggregation
 - Dependencies
 - Using Class Models
- 6. Sequence Diagrams**
 - Sequence Diagrams
 - Interaction Frames
 - Decisions
 - Loops
 - Creating and Destroying Objects
 - Activation
 - Synchronous & Asynchronous
 - The Objects Drive the Interactions
 - Evaluating Sequence Diagrams
 - Using Sequence Diagrams
- 7. Communication Diagrams**
 - Communication Diagrams
 - Communication and Class Diagrams
 - Evaluating Communication Diagrams
 - Using Communication Diagrams
- 8. State Machine Diagrams**
 - What is State?
 - State Notation
 - Transitions and Guards
 - Registers and Actions
 - More Actions
 - Internal Transitions
 - Superstates and Substates
 - Concurrent States
 - Using State Machines
 - Implementation
- 9. Activity Diagrams**
 - Activity Notation
 - Decisions and Merges
 - Forks and Joins
 - Drilling Down
 - Iteration
 - Partitions
 - Signals
 - Parameters and Pins
 - Expansion Regions
 - Using Activity Diagrams
- 10. Package, Component, and Deployment Diagrams**
 - Modeling Groups of Elements – Package Diagrams
 - Visibility and Importing
 - Structural Diagrams
 - Components and Interfaces
 - Deployment Diagram
 - Composite Structure Diagrams
 - Timing Diagrams
 - Interaction Overview Diagrams
- 11. Use Cases**
 - Use Cases
 - Use Case Diagram Components
 - Use Case Diagram
 - Actor Generalization
 - Include
 - Extend
 - Specialize
 - Other Systems
 - Narrative
 - Template for Use Case Narrative
 - Using Use Cases
- 12. Process**
 - Process
 - Risk Management
 - Test
 - Reviews
 - Refactoring
 - History
 - The Unified Process
 - Agile Processes
- 13. The Project**
 - Inception
 - Elaboration
 - Elaboration II
 - Construction Iterations

- Construction Iterations - The Other Stuff
- 14. Domain Analysis**
 - Top View – The Domain Perspective
 - Data Dictionary
 - Finding the Objects
 - Responsibilities, Collaborators, and Attributes
 - CRC Cards
 - Class Models
 - Use Case Models
 - Other Models
 - Judging the Domain Model
- 15. Requirements and Specification**
 - The Goals
 - Understand the Problem
 - Specify a Solution
 - Prototyping
 - The Complex User
 - Other Models
 - Judging the Requirements Model
- 16. Design of Objects**
 - Design
 - Factoring
 - Design of Software Objects
 - Features
 - Methods
 - Cohesion of Objects
 - Coupling between Objects
 - Coupling and Visibility
 - Inheritance
- 17. System Design**
 - Design
 - A Few Rules
 - Object Creation
 - Class Models
 - Interaction Diagrams
 - Printing the Catalog
 - Printing the Catalog II
 - Printing the Catalog III
 - Object Links
 - Associations
- 18. Refactoring**
 - Refactoring
 - Clues and Cues
 - How to Refactor
 - A Few Refactoring Patterns
- 19. Appendix A – UML Syntax**
- 20. Appendix B – Design by Contract**
 - Contract
 - Contracts
 - Enforcing Contracts
 - Inheritance and Contracts
- 21. Appendix C – University Summary**
- 22. Appendix D – Implementations in C++, Java, and C#**