Object Oriented Analysis and Design (OOAD) using UML (TT1130)

Explore OOAD, Use Cases, Best Practices, UML, Patterns, GRASP and More

www.triveratech.com

Course Snapshot

- **Course: Object Oriented Analysis and Design (OOAD) using UML (TT1130)**
- **Duration**: 5 days
- **Skill-Level**: Introductory-level for participants who might specify, design and develop software and applications using traditional/formal/structured methods and want to learn how to apply good practices to OO design and analysis.
- **Hands-on Learning:** This course combines expert lecture, real-world demonstrations and group discussions with pen and paper 'thinking' labs. This is not a coding class.
- **Language / Tools**: This course is programming language neutral and independent of any software development methodology such as waterfall, rapid application development, continuous integration, agile, etc. UML is the primary tool that is used.
- **Delivery Options**: This course is available for **onsite private classroom presentation**, **live online virtual presentation**, or can be presented in a **flexible blended learning format** for combined onsite and remote attendees. Please also ask about our **Self-Paced** / **Video** or **QuickSkills / Short Course** options.
- **Customizable**: This course agenda, topics and labs can be further adjusted to target your specific training skills objectives, tools and learning goals. Please inquire for details.

Overview

Object Oriented Analysis & Design using UML course is a five-day, comprehensive hands-on workshop that will lay a solid groundwork for any developer to easily move into an object oriented programming environment (such as Java, C#, C++, etc)..

This course focuses on the advantages of the OO paradigm and domain modeling in reducing the representational gap between a target domain and the software application itself. Minimizing this gap leads to more effective solutions that are both flexible and robust. The modeling notation taught and used in conjunction with the course is the industry standard UML (Unified Modeling Language). UML provides a programming language independent framework for the analysis, design, programming and testing of software applications. Using a combination of UML and various techniques for analysis and design, the course relates Object Oriented concepts to modeling complex problems. Models built using these techniques have a high success rate when turned into working code.

The course includes coverage of the most effective techniques in use today, such as Use Case analysis, static and dynamic system modeling, responsibility driven design, Design Patterns, using UML to document designs, and much more. The focus of the course is to give a practical approach to producing high quality object-oriented software designs and to provide the knowledge and experience necessary to avoid the most common risks associated with building production systems. Properly assigning responsibilities to classes is one aspect of detailed design that is particularly important. This course pays special attention to that particular design activity, looking at GRASP (General Responsibility Assignment Software Patterns) patterns and how they can be applied to real design problems.

Learning Objectives

Working in an engaging group exercise and hands-on "thinking and drawing" environment, students will:

- Learn the three pillars of building a system; The Model, The Process, The Best Practices
- Have a good, working definition of object-oriented programming
- Understand the object oriented model, including types, objects, encapsulation, abstraction, messaging, protocols, inheritance, polymorphism, relationships, and coupling, strengths and weaknesses
- Understand the concept of representational gap between an application and its targeted domain
- Relate how Domain Modeling minimizes the representational gap between domain and application
- Learn how to read and create the most important UML diagrams
- Recognize the difference between analysis and design
- Be able to produce a requirements analysis

- Know how to create Use Cases, recognizing and avoiding bad use cases
- Effectively perform object discovery using such tools as category lists and use cases to harvest candidate objects
- Learn how to create a static conceptual model of your system
- Learn how to create a dynamic behavioral model of your system
- Understand how to move from analysis to design
- Effectively identify relationships amongst objects, understanding when to show those relationships and when not to
- Effectively assign responsibilities using the patterns and principles of GRASP (General Responsibility Assignment Software Patterns)
- Understand Design Patterns and their importance
- Learn how to apply Design Patterns to refine your model
- Understand the uses of inheritance, where it is appropriate, and where it is not
- Recognize the abuse of inheritance
- Understand the importance and use of interfaces
- Recognize rich versus anemic domain models
- Understand how to move from design to implementation

Course Case Study

Throughout this course students will explore a "real world", practical project illustration (case study) of a typical application showing all the steps required for requirements capture, analysis, architectural and detailed design. The course week begins with a thorough introduction to the fundamental concepts of the object-oriented model and object-oriented programming and moves into in depth coverage of analysis and design techniques, with special emphasis on design patterns. Students will explore the full system lifecycle from initial conception to final delivery.

Students are provided with a clear set of guidelines and rules that they apply to the modeling, from start to finish, of a typical application. These exercises emphasize all aspects of the modeling process with special attention being paid to reusability, extensibility and complexity management plus other techniques that will increase the likelihood that their projects will succeed.

By exploring the lab Case Study students will learn to:

- Understand the Object Oriented Paradigm
- Know how use UML diagrams for modeling systems
- Use the Unified process to guide the analysis and design of a system
- Use Actors and Use-Cases to drive requirements capture
- Build analysis models
- Evolve the analysis model into a complex component-based architectural model
- Use iterative round trip analysis and design techniques
- Know how to verify "goodness" by applying a set of rules and guidelines

Audience & Pre-Requisites

This is a **beginner** level programming course, designed for developers or technical managers who specify, design and develop software and applications using traditional/formal/structured methods and want to learn to use an object-oriented approach.

Attendees can include systems and software analysts and designers, programmers who read and implement program designs, personnel involved in inspections and design/code walk-through, software project managers managing large (re-use) projects, and maintenance personnel involved in maintaining and re-engineering software products. This course is also highly beneficial for those who specify requirements and business rules for systems. Attendees should have a working knowledge of developing software applications.

Take After: Our core OO training courses provide students with a solid foundation for continued learning based on role, goals, or their areas of specialty. Our object oriented developer learning paths offer a wide variety of follow-on courses such as:

- Basic level coding courses in OO technologies such as Java, .Net, C++, C#, Python or JavaScript
- Software development, Architecture or Design Patterns courses
- Please contact us for recommended next steps tailored to your longer-term education, project, role or development objectives.

Course Topics / Agenda

Please note that this list of topics is based on our standard course offering, evolved from typical industry uses and trends. We'll work with you to tune this course and level of coverage to target the skills you need most. Topics, agenda and labs are subject to change, and may adjust during live delivery based on audience interests, skill-level and participation.

Session: Introduction to OO Analysis & Design

Lesson: Introduction to Modeling and UML
- Three Object-Oriented Themes
- Building Models
- Why Build a Model?
- Notation
- What is UML?
- Domains
- The Process of OO Analysis and Design
- OOAD Process: Requirements Capture; Analysis; Domain Design; Detailed Design; Architectural Design
- OOAD Developer Activities
- Activities: Requirements; Object Discovery; Object Relationships; Object Interactions; Object State; Object Activities; Packaging of Objects; Components and Deployment
- Granularity; Levels of Detail
- Requirements Document
- Typical Requirements Document Example

Lesson: Classes and Objects
- Objects
- Objects Provide a Service
- Rendering Objects
- Abstractions
- Responsibilities and Operations
- Operations
- Rendering Operations
- Messages and Public Interfaces
- Instances
- Classes
- Instantiation
- UML Class and Instance Icons
- Encapsulation
- Discovering Abstractions

Lesson: Relationships
- Introduction to Relationships
- Static Relationships
- Dependencies
- Drawing Dependencies
- Associations
- Class Associations
- Associations are Bi-Directional
- Navigability
- Multiple Associations between Two Classes
- Discovering Associations
- Whole/Part Associations
- Composition
- Lab Discovering Whole/Part Relationships
- Generalization/Specialization Relationships
- Terminology
- Generalizations Hierarchies
- What Gets Inherited?
- Inheritance of Methods and Method Overriding
- Using the Overridden Method's Implementation
- Inheriting Associations
- Abstract Classes
- Multiple Inheritance
- Generalization/Specialization Diagram
- Dynamic Relationships
- Sequence Diagrams
- Property of Sequence Diagrams
- Sequence Diagrams Example
- Creating a UML Sequence Diagram
- Communication Diagrams
- Communication Diagram Details
- Creating a Communication Diagram

Lesson: States and Activities
- State Machine Diagrams: Object Lifecycles
- Definitions
- States
- State Machine Diagram Event Triggers
- Entry and Exit Actions
- Activity
- Statecharts Model a Single Object
- Creating a Simple State Machine Diagram
- Activity Diagrams
- Activity Diagram: Similar Notation to State Machine Diagram; Sequence of Steps; Branching; Merging; Synchronization
- Creating an Activity Diagram

Lesson: UML Diagrams
- Introduction
- Class Diagram
- Use Case Diagrams
- Interaction Diagrams
- Sequence Diagrams
- Communication Diagrams
- State Machine Diagrams
- Activity Diagram
- Implementation Diagrams

Session: Object-Oriented Analysis

Lesson: Use Cases
- Where We Are
- Useful Systems
- Discovering the Use Cases
- The Use Case View
- Actors
- Use Case
- Caveats!
- Extending Use Cases
- Including additional Use Cases
- Generalizations
- Discovering Use Cases

Lesson: Use Case Scenarios
- Scenarios
- Primary and Secondary Scenarios
- Essential and Real Scenarios
- Ranking Use Cases
- Documenting Use Cases and Scenarios
- Summary of Requirements Capture Steps
- Use Case Benefits
- Discovering Use Case Scenarios

Lesson: Conceptual Modeling
- Where We Are
- Introduction to the Analysis Model
- Conceptual Modeling
- Concepts; Concept Category List
- Identifying Concepts
- Mapmaking Principles
- Attributes versus Concepts
- Rules of Thumb
- Specification or Description
- Worked Example: Concepts

- Concepts for the Withdrawal Use Case
- Associations
- Worked Example
- Withdrawal Concepts Diagram
- Adding Attributes
- Worked Example
- Updating the Project Dictionary
- Conceptual Modeling

Lesson: Domain Behavior Modeling

- Where We Are
- Domain Behavior Modeling
- System Sequence Diagrams
- System Sequence Diagram: Details
- Worked Example
- Analysis State Machine Diagrams
- Contracts
- Creating Contracts
- Lab Conceptual Model - Part II

Session: Object Oriented Design

Lesson: Discovering Potential Objects using CRC Cards (Optional)

- Discovering Objects
- Brainstorming for Classes
- CRC cards
- Sample Set of CRC Cards
- Completing the Analysis
- Creating a Project Dictionary
- Using CRC Cards

Lesson: Static Design Concepts

- Introduction
- Visibility of Attributes and Operations
- Multiplicity of Objects
- Interfaces and Components
- Design Complex Systems from Components
- Identifying "Good" Classes
- Multiplicity of Associations
- Ternary Relationships
- Alternatives to Ternary Relationships
- Constraints in Associations
- Role and Role Names
- Association Qualification
- Association Class
- Whole/Part Associations
- Extensibility Mechanisms
- Generalization/Specialization - Abstracting Down; Abstracting Up
- Abstract Classes
- Multiple Inheritance
- Types and Substitutability
- Polymorphism
- Specialization is not Always Appropriate!
- Organizing Classes into Packages
- Using Packages
- Component Diagrams
- Interfaces
- Deployment Diagrams

Lesson: Dynamic Design Concepts

- Introduction
- Interaction Diagrams
- Sequence Diagram Example
- Communication Diagram Details
- State Machine Diagrams and Business Rules
- Verifying Completeness
- Advanced States and Transitions
- Superstates and Substates
- Concurrent States
- Creating a Sophisticated State Machine Diagram
- Activity Diagrams: Swimlanes
- Creating an Activity Diagram

Lesson: Domain Design

- Where We Are
- Conclusion of the Analysis Phase
- Starting the Design Phase
- Iterative Development
- Domain Design
- Detailed Design
- Forming the Architectural vision
- Describing Real Use Cases
- Real Use Case: Worked Example
- Review of the Conceptual Model
- Responsibilities
- General Responsibility Assignment
- Patterns
- Open Close Principle
- Dependency Inversion Principle
- Interface Segregation Principle
- Single Responsibility Principle
- Liskov's Substitution Principle
- Golden Rules
- Low Coupling Examined
- Authenticate Use Case: Possible solution
- Creating the Domain Design Model

Lesson: Detailed Design

- Where We Are
- Design Overview
- Detailed Design Steps
- Steps in Checking for Completeness
- Detailed Design Activities
- Class Design
- Ensuring Low Coupling
- Patterns In Design
- Model/View/Controller Pattern
- Factory Pattern
- Proxy Pattern
- Representing Patterns
- Mapping to Databases
- Mapping to User Interfaces
- About Frameworks
- Legacy Data
- Designing Components and Interfaces
- Creating the Domain Detailed Design

Session: Summary & Conclusion

- Usage of OO Technology
- Methodologies and Notation
- Management Issues
- The Unified Software Development Process
- Using Risk to Order the Process
- Implementation Timetable
- Reuse
- Education and Mentoring
- Training and Mentoring

Student Materials: Each student will receive a **Student Guide** with course notes, code samples, software tutorials, diagrams and related reference materials and links (as applicable). Our courses also include step by step hands-on lab instructions and and solutions, clearly illustrated for users to complete hands-on work in class, and to revisit to review or refresh skills at any time. Students will also receive the project files (or code, if applicable) and solutions required for the hands-on work.

Classroom Setup Made Simple: Our dedicated tech team will work with you to **ensure your classroom and lab environment is setup, tested and ready to go** well in advance of the course delivery date, ensuring a smooth start to class and seamless hands-on experience for your students. We offer several flexible student machine setup options including **guided manual set up** for simple

installation directly on student machines, or **cloud based / remote hosted lab solutions** where students can log in to a complete separate lab environment minus any installations, or we can supply **complete turn-key, pre-loaded equipment** to bring ready-to-go student machines to your facility. Please inquire for details, options and pricing.

For More Information

Need dedicated training? All courses can be presented **onsite** or **online,** or in a **combined / flex / blended learning format**, tailored to target your specific audience, needs and learning goals. We also offer focused, flexible **short courses, self-paced learning** options, **recorded sessions** and more. We train beginner to advanced skills in all areas we cover, and offer **New Hire / Cohort Training**, **Boot Camps**, **Skills Immersion Programs**, **Reskilling Programs**, **Skills Migration & Transition Programs**, and more. We collaborate with you to ensure all courses are truly targeted to meet your specific needs and learning skills, maximizing your valuable training time, as well as your important budget.

Please also visit our extensive **Public Training Schedule** for training for smaller groups or individuals. Please contact us for course details, **Corporate Rates** and **Special Discount Offers**.

For more information about our dedicated training services, collaborative mentoring services, courseware licensing options, courseware development services, public course schedule, training management services, partner and reseller programs, or to see our complete list of course offerings and special offers please visit us at **www.triveratech.com**, email **Info@triveratech.com** or call us toll free at **844-475-4559.** Our pricing and services are always satisfaction guaranteed.
