LP:100: Applying Object-Oriented Analysis & Design Using UML 2.0 (5 days)

The Object Oriented Analysis & Design using UML 2.0 course is a five-day, comprehensive hands-on workshop that will lay a solid groundwork for any developer to easily move into a Java programming environment. This course takes advantage of several of the new features in UML 2.0.

Enterprise application developers today must be able to build, evolve and maintain very complex software solutions. Component-based technologies, such as the Object Oriented software development paradigm, are ideally suited to the development of this type of software. This course provides the techniques necessary to build high quality object-oriented software systems that can fulfill your requirements, can scale in both complexity and capacity, and be more easily understood, extended, and maintained.

Much of the success of this paradigm is due to work of Booch, Jacobson and Rhumbaugh who developed the industry standard UML (Unified Modeling Language) and the Unified (initially named Objectory) Process, both based on "best practices" that have been found to work. These tools provide a framework for the analysis, design, programming and testing of software applications. Using the framework provided by UML and the Unified process, the course teaches the student the Object Oriented concepts plus analysis and design techniques and guidelines for modeling complex problems. Models built using these techniques have a very high success rate when turned into working code.

What You'll Learn

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 using CRC, 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.

Working in a hands-on drawing environment, developers will:

- Learn the three pillars of building a system; The Model, The Process, The Best Practices
- Understand the object oriented model, including types, objects, encapsulation, abstraction, messaging, protocols, inheritance, polymorphism, relationships, and coupling, strengths and weaknesses
- Understand the importance of a development process, and the risks of not having one, or having a bad one
- 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
- 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
- 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
- Understand the importance and use of interfaces
- Understand how to move from design to implementation
- Discuss testing, test plans, the testing lifecycle and test methodologies.

Course Overview: Hands-On Learning

Throughout this training 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.

All work can be done the old fashioned way – pads, pens and brains - this course can be strictly right brained! However, portions of the work can also be done using some of the commonly available UML tooling (such as those associated with the Eclipse workbench).

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.

Who Should Attend

This is a beginner level programming course, designed for developers who specify, design and develop software and applications using traditional/formal/structured methods and want to learn to use an object-oriented approach. Ideally students should have some working knowledge of a procedural programming language and syntax, such as C.

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.

Pre-Requisites

Attendees should have working knowledge of developing software applications. Designing and analysis experience is also extremely beneficial. This is not a programming class.

Student Materials and Setup Support

Student Materials include a comprehensive Student Guide complete with detailed course notes, diagrams and a copy of the presentation. Step-by-step lab instructions are clearly illustrated for maximum learning.

We're pleased to provide a detailed set up guide for all private or on-site courses, and as much assistance as you require to prepare your students or classroom for the course. Our support personnel and instructors can be contacted for any advice you may require to prepare your classroom and/or students for attendance.

Optional Pre-Testing & Assessment

We work with you to ensure that your resources are well spent. Through our basic pre-testing, we ensure your team is up to the challenges that this course offers. We will work with you to come up with the best solution to ensure your needs are met, whether we customize the material, or devise a different educational path to prepare for this course.
Course Outline

Course Details

Introduction to Modeling, UML and USDP
- Building Models
- Notation
- Domains
- The Process of OO Analysis and Design
- The Unified Software Development Process

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

Relationships
- Static Relationships
- Dependencies
- Associations
- Navigability
- Whole/Part Associations
- Composition
- Generalization/Specialization Relationships
- Inheritance of Methods and Method Overriding
- Abstract Classes
- Dynamic Relationships
- Sequence Diagrams
- Communication Diagrams

States and Activities
- State Diagrams: Object Lifecycles
- Definitions
- States
- Entry and Exit Actions
- Activity
- Statecharts Model a Single Object
- UML 2.0 Activity Diagrams

UML 2.0 Diagrams
- Class Diagram
- Use Case Diagrams
- Interaction Diagrams
- Sequence Diagrams
- Communication Diagrams

- State Machine Diagrams
- Statechart Diagram
- Activity Diagram
- Implementation Diagrams

Architectural Modeling
- Component Diagrams and Application Architectures
- Component Diagrams and Enterprise Architectures

Use Cases
- Discovering the Use Cases
- Actors
- Use Case
- Caveats!
- Extending Use Cases
- Generalizations

Use Case Scenarios
- Scenarios
- Primary and Secondary Scenarios
- Essential and Real Scenarios
- Documenting Use Cases and Scenarios
- Use Case Benefits

Conceptual Modeling
- Conceptual Modeling
- Concepts
- Identifying Concepts
- Mapmaking Principles
- Attributes versus Concepts
- Specification or Description
- Associations
- Common Association List

Object Oriented Analysis
- Domain Behavior Modeling
- System Sequence Diagrams
- Analysis State Diagrams
- Contracts

Discovering Potential Objects using CRC Cards
- Discovering Objects
- Brainstorming for Classes
- CRC cards & CRC Steps
Static Design Concepts
- Visibility of Attributes and Operations
- Multiplicity of Objects
- Interfaces and Components
- Design Complex Systems from Components
- Identifying "Good" Classes
- Multiplicity of Associations
- Ternary Relationships
- Role and Role Names
- Association Qualification
- Association Class
- Whole/Part Associations
- Extensibility Mechanisms:
  - Abstract Classes
  - Types and Substitutability
  - Polymorphism
  - Packages
  - Using Packages
  - Component Diagrams
  - Deployment Diagrams

Dynamic Design Concepts
- Interaction Diagrams
- Sequence Diagrams
- Interaction Frames
- Communication Diagrams
- Timing Diagrams
- State Diagrams and Business Rules
- Verifying Completeness
- Advanced States and Transitions

Superstates and Substates
Concurrent States
Activity Diagrams: Swimlanes

Domain Design
- Iterative Development
- Domain Design
- Detailed Design
- Forming the Architectural vision
- Low Coupling Examined

Detailed Design
- Detailed Design Steps
- Detailed Design Activities
- Ensuring Low Coupling
- Patterns In Design
- Mapping to Databases
- Mapping to User Interfaces
- About Frameworks
- Designing Components and Interfaces

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