www.iconATG.com

+1 866 ICON US1
+1 636 530 7776
info@iconATG.com

Design Using UML 2.0

The features, principles and techniques of object-oriented technology mitigate the complexities of modern software systems. Successful projects have learned that object-oriented programming is insufficient; that object-oriented analysis, architecture, and design are required for robust, scalable, maintainable web-based and conventional business systems, as well as embedded systems. This course teaches the processes, techniques, and artifacts necessary for modern object-oriented design. Students will learn the chief diagrams, symbols, and concepts of the Unified Modeling Language (UML) v2.0, the de facto international standard for modeling and specifying software systems. UML 2.0 added composite structure, timing and interaction overview diagrams, and significantly enhanced class, component, sequence, and state machine diagrams. Students will learn through detailed lecture and hands-on labs the core competencies in object-oriented design through the use of UML 2.0 diagrams and semantics.

Objectives:

- Master the following UML 2.0 diagrams using proven analysis and design methods:
  - Class diagrams
  - Sequence diagrams
  - Component diagrams
  - State Machine diagrams
  - Composite Structure diagrams
  - Timing diagrams
- Learn and apply key principles that facilitate repeatable, quality designs such as the Liskov Substitution Principle, the Law of Demeter, the Information Expert Principle, High Cohesion, Loose Coupling, and many others
- Understand the basic concepts of object-oriented software architecture including logical partitioning of systems into layers and subsystems, process and thread architecture, and hardware architecture modeling
- Learn what analysis, architecture, and design patterns are and apply them to improve designs
- Acquire hands-on experience in these methods and diagrams through case study exercises

Audience:

This course is designed for systems analysts, architects, designers, developers, and testers who develop object-oriented systems. Technical leads and software quality assurance personnel who oversee development of object-oriented systems will find this course vital.

Perquisites:

Analysis Using UML 2.0

Duration:

3 days

www.iconATG.com

+1 866 ICON US1
+1 636 530 7776
info@iconATG.com

Outline:

1. **Analysis: How do we do analysis?**
   - Identify step in the analysis process
   - Review Domain Class Models, System Sequence Diagrams, System Operation Contracts, and State Models

2. **Architecture: How do we architect a system?**
   - Explore the 4+1 view of architecture
   - Layering and subsystems
   - Component modeling on Component Diagrams
     - Components
     - Provided interfaces
     - Required interfaces
     - Dependencies

3. **Design: How do we design a solution?**
   - Learn the steps in the design process
   - Move from domain modeling to software modeling

4. **Design: How do we design classes?**
   - Learn how to identify software classes
   - Refactor classes to enhance cohesion using
     - Delegation to helper classes
     - Model-View-Controller pattern
     - Parts, ports, and connectors on Composite Structure Diagrams
   - Lab: Model entity, control, and boundary classes in a design
   - Write class specifications
   - Apply architectural decisions such as layering
   - Lab: Apply a layered architecture to a design

5. **Design: How do we design associations?**
   - Learn the full syntax of associations, aggregations, compositions, generalizations, realizations, and dependencies
   - Design role visibility and navigability
   - Design inheritance and polymorphism in generalization relationships
   - Learn the Liskov Substitution Principle
   - Lab: Model relationships in a design class model

6. **Design: How do we design attributes?**
   - Learn the full syntax of attributes
   - Discuss visibility, data typing, multiplicity, class-scope attributes
   - Design derived attributes
   - Write attribute specifications

7. **Design: How do we identify operations?**
   - Learn the syntax and semantics of detailed Sequence Diagrams:
     - Object, activation, creation, and destruction
     - Messages and operation signatures
     - Loop, alternative (alt), option (opt), break & parallel (par) fragments and continuations
     - Interaction use (ref) fragments, decomposed lifelines & gates
     - Critical region fragments
     - Negative (neg), assertion (assert), strict, ignore, and consider interaction operators
     - Part decomposition
   - Discuss process of designing object collaboration through sequence diagrams
   - Guidelines to promote loosely coupled, highly cohesive designs
   - Lab: Develop sequence diagrams

8. **Design: How do we design operations?**
   - Learn the full syntax of operations

- Discuss visibility, parameter and return data typing, class-scope operations
- Design operation types: implementer, manager, accessor, and helper
- Enhance design resiliency by applying the Law of Demeter
- Write operation specifications
- Lab: Refactor sequence diagrams and classes to apply the Law of Demeter

9. Design: How do we model object state?

- Learn State Machine Diagram syntax and semantics:
  - States, Transitions, Events, Activities, Actions, Guards, Constraints
  - Submachine & composite state entry & exit points
- Apply Timing Diagram semantics to model timed state transitions

10. Design: How do we use patterns?

- Define patterns
- Explore representative patterns: Observer, State, Strategy, Abstract Factory
- Learn patterns for all design activities: analysis, architecture, design, and coding
- Modeling patterns and pattern use with Composite Structure Diagrams
- Lab: Apply patterns to a design