Applying OOAD using UML2.0

Course Summary

Description

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) 2.0. 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 very high success rate when turned into working code.

Objectives

At the end of this course, students will be able to:

- 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
- Produce a requirements analysis
- Create Use Cases, recognizing and avoiding bad use cases
- Perform object discovery using such tools as category lists and use cases to harvest candidate objects
- Create a static conceptual model of your system
- 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
- 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

Applying OOAD using UML2.0

Course Summary (cont'd)

Topics

- Introduction to Modeling and OOAD
- Classes and Objects
- Relationships
- Use Cases
- Use Case Scenarios
- Conceptual Modeling
- Domain Behavior Modeling
- Discovering Potential Objects using CRC Cards
- Static Design Concepts
- Dynamic Design Concepts
- Domain Design
- Detailed Design
- Summary & Conclusion
- Remaining UML 2.0 Diagrams
- States and Activities

Audience

This course is designed for developers who 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.

Prerequisites

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

Duration

Five days

Applying OOAD using UML2.0

Course Outline

ProTech
protechtraining.com

"Charting the Course ...

... to Your Success!"

Applying OOAD using UML2.0

Course Outline (cont'd)

IX. Dynamic Design Concepts
- A. Sequence Diagrams
- B. Business Rules
- C. Verifying Completeness
- D. Advanced Sequencing
- E. Concurrent Sequences
- F. Activity Diagrams: Swimlanes

X. Domain Design
- A. Iterative Development
- B. Domain Design
- C. Detailed Design
- D. Forming the Architectural vision
- E. Low Coupling Examined

XI. Detailed Design
- A. Detailed Design Steps
- B. Detailed Design Activities
- C. GRASP patterns/principles revisited
- D. Controller
- E. Creator
- F. Information Expert
- G. Law of Demeter
- H. Low Coupling/High Cohesion
- I. Polymorphism
- J. Pure Fabrication
- K. Good/Bad packaging principles
- L. Coupling (allowed and/or required communication paths), layering, and dependencies
- M. Patterns In Design
- N. Mapping to Databases
- O. Mapping to User Interfaces
- P. About Frameworks
- Q. Designing Components and Interfaces

XII. Summary & Conclusion
- A. Usage of OO Technology
- B. Methodologies and Notation
- C. Management Issues
- D. Reuse

XIII. Remaining UML 2.0 Diagrams
- A. Use Case Diagrams
- B. Interaction Diagrams
- C. Communication Diagrams
- D. Sequence vs. Communication Diagrams
- E. State Machine Diagrams
- F. Statechart Diagram
- G. Activity Diagram
- H. Implementation Diagrams

XIV. States and Activities
- A. State Diagrams: Object Lifecycles
- B. Definitions
- C. States
- D. Entry and Exit Actions
- E. Activity
- F. Statecharts Model a Single Object
- G. Analysis State Diagrams
- H. Activity Diagrams: Swimlanes