

# In-depth Object Orientated Analysis and Design with UML

## **Duration**

4 days

## **Instructor**

Steve Adolph

## **Class Limit**

20 students

## **Prerequisite**

It is strongly recommended that participants be familiar with basic OO concepts such as objects and classes.

## **Price**

On-site:

Please contact SPC for pricing

Public Training:

\$1980 (4 days)

\*Discount available for early registration

## **Materials Provided**

- Student manual containing the course slides
- Student handouts with class exercises and class studies

This seminar will provide practical guidance and hands on experience in the design of object oriented software using the Unified Modeling Language (UML). At the start of the seminar, participants will receive specifications for a system and will follow the instructor through the steps of planning, analysis, design and implementation for the project.

## **Intended Audience**

This seminar is intended for analysts, software developers, and senior managers with technical background who are interested in a rapid overview of the object oriented analysis process using UML.



## **Instructor**

Steve Adolph is a principal with WSA Consulting Inc. He consults in the area of object technology including Use Case consulting, patterns, UML, and the software development process. Steve has been mentoring local and US clients to migrate their software development process to include the appropriate level of object technology. Steve Adolph received his Masters in Computing Science from SFU in 1987.

TRAINING

## In-depth Object Orientated Analysis

### Outline

#### *Projects, Risks and Methodologies—Why Do Good Projects Fail?*

Risk is the enemy of all software projects and successful software projects reduce risk to manageable proportions. Object oriented software development promotes an iterative style of software development that aids software risk management.

- Identifying and managing risks
- Understanding how complexity contributes to risk
- How a methodology helps to cope with risk?
- How OOA/D enables an iterative software development lifecycle?

#### *Software Development Life-Cycle—Managing the Project*

The software development lifecycle specifies the ordering of the steps in a methodology and the major milestones.

- Iterative lifecycle models
- Unified process and other iterative software development processes
- Planning an object oriented project using an iterative software development lifecycle

---

For more information on this or other SPC Springboard courses, please visit [www.spcspringboard.com](http://www.spcspringboard.com) or e-mail SPC at [info@spc.ca](mailto:info@spc.ca)

Software Productivity Center  
Suite 460—1122 Mainland Street  
Vancouver, BC V8M 4T8  
[www.spc.ca](http://www.spc.ca)

Toll Free:  
1.877.548.1948

Fax:  
604.689.0141

Vancouver:  
604.662.8181

Toronto:  
416.885.0512

---

#### *Requirements Capture—Understanding the Needs of the User*

A Use Case is the story of how an Actor uses the services provided by the system. Use Cases provide an effective technique for capturing requirements and for managing a software development project.

- Actors and Use Cases
- UML notation for Use Case models
- Communicates includes and extends associations
- High level Use Case model
- Use Cases and scenarios
- Describing Use Cases with scenarios
- Expanding the Use Case model
- Use Case specification
- Styles for writing Use Case specifications
- Use Cases Model and the software requirements specification
- Use Case driven project management

#### *Deliverables*

During this section participants will create or receive copies of the following deliverables:

- High level Use Case model
- Project plan
- Expanded Use Case model
- System scenario diagrams
- Event list
- Software requirements specification

#### *Requirements Analysis—The UML Notation*

The UML Notation is rapidly becoming the industry standard notation for describing object oriented systems. It

*Continued on page 3*



## In-depth Object Orientated Analysis

### *Continued from page 2*

has a large repertoire of icons and symbols for the expression of relationships, objects, class types, and their modifiers. Participants will learn how to use the UML notation to create concise models for their systems.

- Class Diagrams
- Object Diagrams
- Packages
- Interaction diagrams

### *Domain Analysis*

Domain analysis is concerned with devising a precise, understandable, and correct model of the real world. Object oriented analysis provides the designer with an understanding of the problem in preparation for design. Participants will learn a variety of methods for discovering and understanding the classes that exist in the application domain.

- Object discovery
- Identifying relationships
- Defining operations
- Identifying attributes
- Re-using abstractions with inheritance
- Applying patterns to recurring problems

### *Deliverables*

During this section participants will create or receive copies of the following deliverables: · Revised project plan · Class list · Use Case realization

### *Patterns—Rules of Thumb*

A pattern is a solution to a problem in a context, effectively a rule of thumb. A set of patterns known as the General Responsibility Assignment Patterns (GRASP) can assist the designer when creating object oriented systems.

- History of patterns
- GRASP patterns
- Design patterns
- Patterns for object oriented software development

### *Design—System Design*

During system design, the architecture of the system is chosen. The architecture establishes common tactical policies that must be used by different elements of the system. Participants will learn which

aspects of an application problem they should consider when formulating a design.

- Organizing the system into subsystems
- Identifying concurrency
- Allocating subsystems to processors and processes
- Selecting data store implementation strategies
- Selecting control strategies
- Establishing trade-off priorities
- Applying patterns to common design problems
- Collect classes into packages and subsystems
- Refine class package structure to reduce coupling and enhance re-usability

### *Object Design*

Object design determines the full definition of the classes and associations used in the implementation as well as the interfaces and algorithms of the methods used to implement the operations.

- Design algorithms to implement operations
- Optimize access paths to data
- Implement software control by refining strategies selected during system design
- Adjust class structure to increase inheritance
- Design implementation of associations
- Determine the exact representation of object attributes

### *Deliverables*

During this section participants will create or receive copies of the following deliverables:

- System architecture document
- System design document



TRAINING

