Object-Oriented Analysis & Design

Goals:

This seminar presents the fundamentals of object-oriented analysis & design for developers of computer based systems. Application Requirements are shown to be developed by a process of analyzing (and modeling) the desired (or existing) behavior of an application domain as perceived by domain experts (Users) working in that domain. From this domain requirements models, attendees are shown how to create an object-oriented specification of a computer-system capable of providing the behavior required by the expert user in the application domain.

During object-oriented design stage of system development, User requirements are mapped to application software-classes in an appropriate OO programming language. Technology software-classes are added to handle the User interface, databases for ensuring the persistence of application-objects, communications between different processors and processes and mechanisms associated with the operation of the computer (e.g. processes, interrupts etc.). The object oriented approach. is capable of modeling any business system, industrial process, natural system, computer system or office work flow.

Duration:

3 full days; normally 9 a.m. - 5 p.m. daily.

Breaks:

5 min. each hour, 15 min. mid-morning, 1 hour for lunch and 15 min. mid-afternoon.

Format:

Students receive daily instruction in the concepts, rules and techniques of object-oriented analysis and design.

Students break up into groups of 2-3 to work on exercises provided from an appropriate case study selected for the seminar. Each exercise follows the introduction of a major new topic and acts to reinforce the class room instruction with a practical application.

Attendees:

Those involved in building systems that must generate a planned of response as a result of inputs from its environment. Engineers constructing process control systems, data acquisition systems, embedded controller systems, distributed MIS systems, client/server systems, online/interactive/real-time system etc. would benefit from this seminar.

Handouts:

Each student receives a binder containing a paper copy of the instructor's overhead transparencies.

Case Study:

One of several case study examples could be chosen for a public seminar based on background information provided by students on their registration forms.

For in-house seminars, new case studies can be developed to match the expectations of the host company. A three week (minimum) confirmation period is required to develop a new case study. The host company should provide detailed background material.

Detailed Seminar Outline

Part 1 - Introduction to Systems Development

The first part of the seminar introduces students to the notion of systems, their generic makeup and the idea of a systems-development methodology. In addition the idea of a systems development life-cycle and the need for appropriate project management is introduced.

Chapter I - Developing Systems in an OO Manner

This chapter reviews the seminar and introduces the student to the goals of object-oriented system development. The advantages of the object-oriented approach are reviewed and its use on typical projects presented.

Chapter II - Developing Intelligent Systems

The notion of a system development methodology are presented and different approaches to developing systems are described. Several system development life-cycles are introduced and the role of a project manager in managing the system development within a life-cycle are discussed.

Part 2 - Introduction to Domain Modeling

The second part of the seminar introduces both the foundational concepts of the object-oriented paradigm and the importance of adopting a modeling approach to systems development. The importance of graphical models is highlighted. The central role played by humans in the model building process is explained and the basics of modeling introduced. Finally, the many different approaches to modeling in a application domain are discussed and mechanisms are introduced to limit the modeling activity in a domain and prevent "analysis paralysis".

Chapter III - Human Factors

This chapter briefly introduces the student to the ways in which humans approach problem solving. The human disposition for pattern recognition and language are introduced and the well known limits to human cognition are presented. The importance of these concepts are stressed for managers choosing an OO development methodology.

Chapter IV - Modeling Theory

The fundamental concepts of OO model building are presented in this important chapter. The nature of form, abstraction and structure are described and the most general definition of a “class” - as a form perceived by humans - is introduced. relationships between objects are introduced and the notion of sub-type and super-type classification is presented. Modeling notations are introduced and several well-known modeling activities are discussed.

Chapter V - Modeling in a Domain

The fundamental ideas of "domains" and "systems" are introduced and defined precisely. The advantages of modeling at different levels of abstraction are introduced and the idea of setting a particular modeling viewpoint are discussed. The notion and contents of a development charter are presented as a way to control the scope and depth of model development in a domain.

Part 3 - Modeling Domain-Structure

Part 3 is the first of three parts describing, in detail, how to model in an object-oriented manner within a given domain. Attendees are shown how the static-structure of a domain is described in terms of the objects found within that domain and the relationships between those objects. The "structure" of objects is defined in terms of their attributes and finally the students are shown how to create an object model describing the objects within a domain.

Chapter VI - Objects in the Domain

The analytical skills of finding objects in an application domain from a description of that domain are described. The Object Model notation is introduced as a way to represent objects and relationships in a domain. A simple table representation is shown to be useful for pragmatically discussing details of object-instances.

Chapter VII - Associations in the Domain

The analytical problem of finding the allowed relationships between object-instances from the facts contained in a description of the application domain is described. The equal importance of relationships in an OO model is emphasized. The Object Model representation of these relationships is presented and the many different forms of relationships and their representation on the Object Model is discussed. Similarly, a table representation of relationships is presented to match those introduced earlier for objects.

Chapter VIII - Object Structure

The structure of an object, in terms of its properties, is introduced. Associative-objects are discussed and the concept and meaning of aggregate objects described. Students are taught to recognize the differences and similarities between these two types of objects from an analysis of their properties.

Chapter IX - Classification of Objects

The human ability to classify object instances into classes is reviewed. Further analysis of a classified set of the instances into smaller sub-classes or the possible synthesis of existing classes into a larger super-class are discussed. This multi-way classification technique is presented as a natural mechanisms for capturing knowledge about objects in a domain. Both the Object Model and the table representation of the object classification hierarchy are presented.

Chapter X - Creating a Domain Object Model

The practical creation of an Object Model is presented in this chapter utilizing the approaches described in earlier chapters. The domain description (or expert testimony) is shown to be the foundation of all techniques. The steps to be followed in creating an Object Model are presented. Common problems (and their solutions) with draft Object Models are discussed along with the constraints due to different types of associations between objects.

Part 4 - Modeling Object Behavior

Part 4 introduces the ideas of applying State Transition Diagrams (STDs) to specifying object behavior. A method of creating and validating STDs is introduced.

Chapter XI - Behavior

The concept of object behavior is introduced. Modes of behavior are shown to be the primitive units of behavior we use to describe changes in the world around us. The causes of transitions between the modes are described. The three classical forms of STDs are reviewed and the combined form used throughout the seminar described. The terminology of “state” or “status” for “mode of behavior” is presented. The simple connection between the way we describe behavior in English and the notation of STDs is presented.

Chapter XII - Application of STD's to Modeling Class Behavior

The detailed notation used in STDs to specify modes of behavior and transitions between the modes are reviewed. Several example behaviors are presented to illustrate the notation. The discovery of events that cause the creation and removal of object instances in a domain is discussed along with those events that cause the properties of an instance to change. Similarly the importance of noting the events that cause the object instance to form or break relationships to other objects is highlighted.

Chapter XIII - Creating an Object STD

The practical matter of creating the behavior model of an object is presented in this chapter. The domain description is shown to be the foundation of the work to create a draft STD for each object. Common problems to watch for in a draft STD are discussed and methods for their elimination recommended. A summary of the STD notation is provided for completeness.

Part 5 - Modeling Interactions

Part 5 synthesizes the static state description of domain objects, provided in Part 2, with the dynamic state description of domain objects presented in Part 3. The importance of processes that provide the object's external behavior and act to change its properties are introduced. A formal notation is presented for their representation in an OO process model of each class. This section emphasizes the object-oriented approach to specifying object behavior by its defined actions and activities and contrasts it with the more primitive, but popular, non-OO functional process model of Use-Cases.

Chapter XIV - Process Structure of Objects

Object Structure Diagrams are introduced to represent the "internals" of an object. These diagrams are an extended form of data flow diagram, familiar to most system developers. The diagrams are shown to contain representations of the objects behavior, its properties and the processes that cause both the behavior and properties to change. In addition, students are introduced to the notations for events, conditions, data flows, energy flows and mass flows that are required to carry out the processes required of real-world objects. The latter are of interest to real-time system developers and simulations of the real-world.

Chapter XV - Interaction Between Objects

Object Interaction Diagrams (OIDs) are introduced, as enhanced data flow diagrams, to represent interactions between objects. Students are shown how to complete OIDs following the completion of all Object Structure Diagrams. Energy, mass and information flows required by some objects are shown to be created by other objects in the domain. The concept of leveling OID's is introduced and guidance is provided for the leveling process.

Part 6 - Intelligent-System Specification

Part 6 introduces students to the important notion of systems that interact with their immediate environment by accepting information inputs, producing information outputs and exhibiting intelligent behavior. Humans, computer based systems and some dedicated hardware systems qualify as intelligent systems. The process of systems' specification (traditionally called just systems analysis) following domain analysis is shown to be invaluable to system engineers constructing interactive, online or real-time systems. The process of deriving the system specification from the domain requirements is fully described.

Chapter XVI - Deriving New Intelligent-Systems

The steps leading from the identification of a problem in a domain to its solution, and the production of a domain model of the solution are reviewed. Intelligent systems are defined and introduced as the active components of a domain that automate the solution of many problems. Students are shown how the specification of an intelligent system results in the "banishing" of all energy and mass flows into the terminator objects. Intelligent system interact with these terminator objects only by the exchange of event and data flows.

Chapter XVII - Establishing the Intelligent-System Context Diagram

Students are shown how to develop the traditional Context Diagram for an intelligent system from the earlier developed domain models. The need to define the flows to and from the terminator objects is discussed. Guidance is provided for those system builders who may not have time to create a set of domain models.

Chapter XVIII - Establishing the Intelligent-System Structure

The procedures for deriving the final system-object structure of the intelligent system are presented. Similarly the behavior of each system-object is derived from the behavior of the original domain models of the same objects. For distributed and real-time system builders the design of the external system equipment (terminator objects) is shown to influence of the final behavior of the system-objects. Students are show how to reduce the Object Interaction Diagram to the system's Object Communications Diagram (OCD) in which the only allowed interactions between system-objects is due to the exchange of data flows. Finally, leveling of the OCD is introduced.

Chapter XIX - Specification

The nature of requirements and specifications are reviewed. The object-oriented modeling techniques introduced in the seminar are shown to provide both a model of the system requirements and a specification of a system to automate the provision of those requirements. A "road-map" of model development is presented and the structure of a completed requirements specification described. Finally, to close off the discussions of analysis, students are introduced to the notions of non-functional requirements such as reliability.

Part 7 - System Design

The final section of the seminar discusses the important topic of object-oriented design. During the design phase, the user's requirements specification is translated into the specification of a set of software classes that are capable of providing the required functionality and behavior. Techniques for making this software design easy to maintain and re-use are described.

Chapter XX - System Architectural Design

The notions of "design" is discussed and defined for this final section of the seminar. The process of architectural design, wherein requirements are allocated processors, processes, dedicated hardware and human-staff is introduced. Simple allocation schemes are discussed for single and multiple processor hardware architectures.

Chapter XXI - Selecting a Software Environment

The software environment is introduced as the set of software that provides all system services to the application programs e.g. User interface, dbms, communications, security etc. In modern systems this software is often present in the form of object-oriented API libraries. Software environments are shown to be generic and re-usable between different applications in very different domains. Selection criteria are reviewed for business and real-time systems. The idea of modeling the software environment is introduced and its importance for real-time systems stressed. These models are known as frameworks or pattern in today's OO parlance.

Chapter XXII - Software Object Design

The problems to be solved in designing software classes are reviewed. Given the existing object oriented programming languages, the remaining major design problems are to implement object behavior, associations between classes and persistence of properties and associations.

Chapter XXIII - OO Programming Language Issues

The structure of modern OO programming languages are reviewed. These features are the only possible mechanism that a programmer can use to implement the software specification. Language features such as messages, methods, inheritance and memory management are presented in a language independent manner.

Chapter XXIV - Establishing a Design Mapping

The most reliable and maintainable approach to producing software from a specification is by the adoption of a set of design rules for the programming team. These rules tell programmers how to translate specification models into code (except the algorithms). Simple rules for the implementation of classes, properties, associations, behavior etc. are described. This approach is show to simplify class testing.

Essentials of Hands-on OO Programming in Java

Goals:

This hands-on seminar presents the essentials elements of the Java object-oriented programming language. Java is a relatively new OO language and far simpler than C++ to learn. The seminar provides instruction in a simple approach to object-oriented analysis with the design and coding of the analysis models in the Java language. This combined approach has worked well for the instruction of the Smalltalk and Objective-C languages due to their very simple syntactic structure. During the seminar, students are introduced to Java language as they follow through the analysis, design and coding of a simple case-study example. The approach combines the abstraction in analysis with the practical needs of coding.

Students are exposed to the syntax of the Java language as they analyze and code Java applets and applications in a hands-on lab environment. The core classes, input and output, exceptions, threads and graphical programming are presented. After this seminar, students are well positioned to undertake an OO projects from beginning to the end.

Duration:

4 full days; normally 9 a.m. - 5 p.m. daily.

Equipment:

Computers are required; 1 for every 2 Students.

Breaks:

5 min. each hour, 15 min. mid-morning, 1 hour for lunch and 15 min. mid-afternoon.

Format:

Students receive daily instruction in the concepts, rules and techniques of object-oriented analysis, design and coding syntax of the Java language.

Students break up into groups of two to work hands-on on exercises provided from an appropriate case study selected for the seminar. Each exercise follows the introduction of a major new topic and acts to reinforce the class room instruction with a practical application.

Attendees:

Those involved in building systems that must generate a planned of response as a result of inputs from its environment. Engineers constructing process control systems, data acquisition systems, embedded controller systems, distributed MIS systems, client/server systems, online/interactive/real-time system etc. would benefit from this seminar.

Handouts:

Each student receives a binder containing a paper copy of the instructor's overhead transparencies.

Case Study:

One of several case study examples could be chosen for a public seminar based on background information provided by students on their registration forms.

For in-house seminars, new case studies can be developed to match the expectations of the host company. A three week (minimum) confirmation period is required to develop a new case study. The host company should provide detailed background material.

Detailed Seminar Outline

Introduction

The seminar contents and the approach to be followed is described. The time of breaks and start and stop times are confirmed with the students.

Chapter I - The Java Niche

The position of Java in the world of OO system development is described. Its strengths and weaknesses are reviewed and the niche it is expected to fill described. The Internet and the World Wide Web are described. The security problems associated with surfing the Web are reviewed.

Chapter II - Hello Web-World

A simple, short program that prints “Hello World” across the Internet is introduced to test the hands-on facilities at the Customer’s site and to introduce Students to running their Java development systems. Code creation, code management, editing, compiling, executing and debugging are undertaken with a simple example.

(If problems arise a local contact should be available to remedy the situation.)

Chapter III - Introduction to OO

A history of OO is presented from an analysis and an implementation perspective. The benefits of adopting an OO approach are shown to flow from a defined process for building automated systems (i.e. those in which computers follow instruction rather than humans). Students are introduced to the manner in which humans describe the world and record it in the English language, and how this is subsequently translated into a computer language.

Chapter IV - Introduction to Java

The Java compiler, code-package “libraries” and the Java virtual (interpreter) machine are introduced. The notion of “just-in-time” local compilation of the byte-codes is described. Java is compared to the other major procedural OO programming languages C++, Smalltalk and Objective-C. The benefits and drawbacks of a strongly types language are reviewed and its hopes as being (yet another) completely portable language are described.

Chapter V - Objects in Java

The fundamental unit of computation in Java is the object. The manner in which object structure and behavior are defined is introduced and the creation and destruction of objects demonstrated. The life-cycle of an object is discussed. Students are shown how to identify classes of application entities in a problem domain during analysis and represent them (i.e. follow a design procedure) by objects of a Java class.

Topics covered include:
- comment lines
- assignment
- simple “data-types”
- constructors
- messages & actual parameters

instance variables
reference types
methods & formal parameters
computation in OOPLs
garbage collection & finalize

Chapter VI - Introduction to Input & Output

Basic input and output sufficient for the first few chapters is covered.

Topics covered include:
- Streams

Print streams

Chapter VII - Using some System Objects

Students are introduced to using simple system provided classes; to assign them to instance variables and send simple messages to them. The notion of using class libraries is introduced and the documentation standard for the classes and methods described.

Topics covered include:

- Day
- Date
- Strings

Chapter VIII - Classes in Java

Similar to Smalltalk, but unlike C++, classes are objects at run time in Java. The notion of an object representing the entire class of a defined set of objects is discussed and its uses presented.

Topics covered include:

- static variables
- static methods
- classes

Chapter IX - Extending Classes

Creating a sub-class of an existing class is introduced. The design of a class to better allow it to be subclasses is described. The difference between overloading and overriding is discussed.

Topics covered include:

- public, private etc.
- overloading methods
- overriding methods
- final
- designing for extension
- polymorphism & why
- this & super

Chapter X - The Root of all Classes

The Java OO model includes a single class called Object from which all other classes are sub-classes. Students are introduced to its general features and those of the class called Class. The notion of abstract classes and methods are discussed.

Topics covered include:

- class Object and Class
- abstract classes & methods

Chapter XI - Control Flows

The traditional control flow constructs of the C language are presented and their differences in Java described.

Topics covered include:

- statements
- relational & boolean operations
- if/else
- switch

- while & do/while
- for
- labels
- break
- continue
- return

Chapter XII - Arrays

Arrays are presented as one of many types of collection classes. Arrays have a special syntax in Java while the others do not.

Topics covered include:

- declarations
- initialization
- location in class hierarchy
- sizing
- dimensions

Chapter XIII - Implementing Relationships in Java

Students are shown how to identity essential relationships between application domain objects and translate these (via design rules) into collections of objects in Java. The forming and breaking of relationships is discussed and the simple methods required to carry this out in Java introduced. The use of relationships during the creation of a User inquiry is presented.

Chapter XIV - Building Object Behavior in Java

Students are shown how to identity essential modes of behavior in application objects and translate this (via design rules) into public and private methods in Java. The behaviors are often known as “states” or “statuses” in the application domain.

Chapter XV - Interfaces

Java does not provide multiple inheritance of classes (as in C++) but does provide for multiple inheritance of a (class) interface. The definition of an interface in terms of its methods is described and their use reviewed.

Topics covered include:

- multiple inheritance
- interfaces
- implementing an interface
- the Run interface
- when to use interfaces

Chapter XVI - Java Syntax Review

The formal syntax of the Java language, presented so far by examples, is reviewed.

Topics covered include:

- Character set
- identifiers & declarations
- reserved words
- literals
- Non-OO types
- types of variables
- initialization
- static initialization blocks

Chapter XVII - Typing in Java

The notion of a “type” and a “sub-type” is defined. Classes follow types in that both are defined in hierarchies. The safety issues (i.e. showing nothing bad can happen) in ensuring the typing mechanisms are not breached are briefly discussed. The early Internet security breaches involving types are presented.

Topics covered include:

- type hierarchy
- sub-types
- type casting
- casting rules & safety

Chapter XVIII - Exceptions

Exceptions are a mechanism to ensure that the inputs and outputs of every method are always fully defined. They can be put to other uses where this property is not so obvious. Students are shown how exceptions alter the normal “flow of sequential control”

Topics covered include:

- throws
- try/catch/finally
- when to use exceptions
- non-local control flow
- re-throwing an exception
- exceptions in the interface

Chapter XIX - Java Threads

The Java virtual machine provides the ability to run several threads-of-control within a single Java program. Each thread has a priority, can be started and stopped repetitively, and can preempted by a higher priority thread. The uses of threads are described and Java “synchronized” methods mechanisms for protecting object-properties against being updated “simultaneously” by two concurrent threads reviewed. The problem of deadlock is introduced and students are shown how to create a model from OO analysis that will show the presence of deadlock in a Java application. Methods for eliminating it are discussed.

Topics covered include:

- Creating threads
- synchronized methods
- synchronized statements
- volatile
- thread states
- thread scheduling

- suspending threads
- interrupting threads
- thread groups
- cooperating & selfish threads
- deadlock & its detection
- starvation
- daemon threads

Chapter XX - Applications & Applets

When building an OO application a second OO analysis is carried out (usually only once) by the implementation team to identify those essential parts of an application that must be created (or re-used) for every application built. Examples are databases, interfaces, communications, security etc. These classes of things are provided by frameworks and either purchased or built in-house, once. Students are introduced to the notion of analysis of an implementation domain and shown how to code the solutions and incorporate them into applications.

Topics covered include:

- applications & main
- applets
- applets in HTML
- life-cycle of an applet

Chapter XXI - Developing an Applet's Graphical User Interface

Applets are a Web idea. Code is dynamically loaded over the Web and pressed into execution. Students are shown how to create a simple applet, locate it into a file, embed it into an HTML page and have it execute as the page is accessed over the Web. The parts of the interface are described and students are shown how to layout a page and handle the events from mouse and keyboard.

Topics covered include:

- buttons
- text
- panels
- check boxes
- menus
- layouts
- event driven programming
- "flow" of event handling
- handling events in the GUI
- mapping GUI events to app events
- handling events in the app
- class loading over the Internet
- security

Chapter XXII - Java Packages

Java packages and code management are described. The contents of a few system packages are described.

Topics covered include:

- naming
- access
- contents & location
- "includes"

Chapter XXIII - Intro to Useful Packages & Classes

The basic utility methods needed by applications are covered and the manner in which they are located described.

Topics covered include:

- Vectors   Hash Tables
- math utilities   times & dates

(* Optional) Chapter XXIV - Providing Object Persistence

Persistent objects are software objects who's state is preserved between executions of a given application. They are usually stored on a disk (often in a relational dbms) and brought into memory when required i.e. when they are sent a message from another object related to them. Modern OO database systems make the mechanisms of disk-to-memory-to-disk transparent. The prevalence of relational databases means that Students will, likely, find they are forced to use a relational dbms. The packages available and approaches to persistence are described.

(* Optional) Chapter XXV - Inter-Object Communications

Students are briefly introduced to the notion of system architectural design. In this step students are shown how to partition an application between different computers. The communications between objects in different computers is then discussed and the various technologies available today (e.g. CORBA, sockets, Java-to Java, etc.) are reviewed.

Chapter XXVI - Web Information Sources on Java

If possible, students connect to the Internet and "surf" to the well known locations that provide information about Java.

Learning Objectives for OO Analysis & Design

Students will learn the following:

Students will learn the history of OO from its early theory to present day practice.

The very large difference between carrying out analysis and carrying out design tasks will be discussed in detail to ensure students know when to "listen" on a project and when to make decision.

The four standard OO models used to represent both OO analyses and OO designs (i.e. EERDs, STDs, OSDs and OIDs) are presented and students taught how to create them from interviewing Domain experts or from reading a description of a domain.

Students are taught how Extended Entity Relationship Diagrams (EERDs) represent the essential entities, and relationships between them, in the problem domain. This model is later "translated" during OO design into Object Models and finally into an OO programming language.

State Transition Diagrams (STDs) are introduced as a simple way of describing how the entities change their state and status during their life-cycle from birth to death. The basis of event-based programming is discussed as the method whereby STDs are programmed.

Students are shown how scenarios are developed from the STDs of all the entities. These may be captured by OMT Event Trace Diagrams.

The transition into the implementation phase is introduced by showing the student how to create a Business-System Specification from the Business Requirements modeled above.

Students are introduced to the several phases of implementation covering the allocation of software to various processors (a.k.a. "tiers")and to various processes.

The software design from the Business requirements is introduced as the search for design rules based on a knowledge of the OO programming language, the communications requirements, the object persistence requirements etc. This approach has been taught for 20 years and now finds its way into the popular press as being "pattern" based design.

Learning Objectives for Essentials of Hands-on OO Programming in Java

Students will learn the history of the evolution of the Java language and its potential to become pre-eminent in the new order of the world-wide web.

A simple OO analysis approach will be taught and students will be introduced to OO design directly into the Java language.

The fundamental syntax of the Java language will be presented during the development of a sample case-study project. This hands-on approach reinforces the learning of essential background l material with the practical needs of a developer to learn how to use the features of the language in the most appropriate manner.

Students will learn how and when to make sub-classes of a base class, when to use interfaces and how to navigate their way around the inheritance hierarchy of the Java classes provided by the Sun development kit (the “JDK”).

The notion of inheritance in Java will be taught and the rules involved in overloading and over-ridding super-class methods, and their numerous modifiers, discussed within the context of the case study.

Basic input and output techniques will be introduced: initially for text based printing and later for colorful, event based graphical user interfaces.

Students will be taught how to develop “stand-alone” applications and applets that run from web based HTML pages.

The use of threads in applets (and applications) will be covered and students taught techniques to use threads to create dynamic displays on web pages.

The Java libraries will be explored and certain of the classes employed in the case-study to develop the Students knowledge of Java features “that everyone should know”.
Several advanced topic will be presented covering exceptions, object persistence and inter-object communications to round out the Students knowledge.

Question and answer sessions will be intermingled with class room style teaching and lab-style hands-on project work to allow Students to investigate matters of particular interest to their needs.