Object Oriented Modeling and Design

Notes

OBJECT-ORIENTED MODELING AND DESIGN

Introduction

Contents:

- Introduction.
- Course Relevance
- Learning Outcomes
- Overview of the syllabus
- Introduction to Object Orientation

Introduction

Object Oriented Approach is innovative and modern approach of designing the system by focusing primarily on Data elements of the application domain. It differs from the functional/traditional approach by providing features like data hiding, encapsulation and better reuse.

Modeling is not a separate phase but it is involved in every phase of software engineering. Modeling is all about making models/prototypes of the system/situations needed to do better analysis, design, coding and testing

Prerequisite

The course is aimed at:
Students who have prior knowledge to the concept of Software Engineering and any one Object oriented language like C++.

Course Relevance

Object Oriented Modeling and Design is thinking about the problem using models organized around the real world concepts. Earlier to this was the Procedural oriented paradigm. Today's applications have grown to be very Complex. In order to handle this inherent complexity OOMD was framed. Object Oriented Paradigm addresses the problem Domain by considering the problem as a set of related interacting objects.

The modeling task then is specifying, for a specific context, those Objects (or the Class the Objects belongs to), their respective set of Properties and Methods, shared by all Objects members of the Class.

Design Patterns is a general reusable solution to a commonly occurring problem in software design. A design pattern is not a finished design that can be transformed

directly into code. It is a description or template for how to solve a problem that can be used in many different situations

Learning Outcomes:

At the end of the course the student would have the:

- The Knowledge of the basic concepts of Object oriented modeling and Design.
- Will be able to use the Object Oriented notations and process that extends from analysis through design to implementations.
- Be able to use all the standard UML notations.
- Capable to model the requirements with use cases and describe the dynamic behavior and structure of the design.
- Easily create a modular design with components and relate the logical design to the physical environment.
- The Student will be able to use the concept of design patterns and apply it where suitable.

Overview of the syllabus

The syllabus is divided into eight units covering up the topics of Modeling and design patterns.

Unit I:

- What is Object Orientation?
- What is OO development?
- OO themes;
- Evidence for usefulness of OO development;
- OO modeling history.
- *Modeling as Design Technique:* Modeling; abstraction; The three models.
- Class Modeling: Object and class concepts; Link and associations concepts; Generalization and inheritance; A sample class model; Navigation of class models; Practical tips.

1.1 Terminologies:

- What is Object Orientation: Organization of software as collection of discrete objects that incorporate both data structures and behavior.
- Identity: data is quantized into discrete, distinguishable entities called Objects.

- Classification: means objects with the same data structures and behavior are grouped into a class
- Inheritance: is sharing of attributes and operations among classes based on a hierarchical relationship.

- Polymorphism : same operation may behave differently for different classes. (eg move operation)

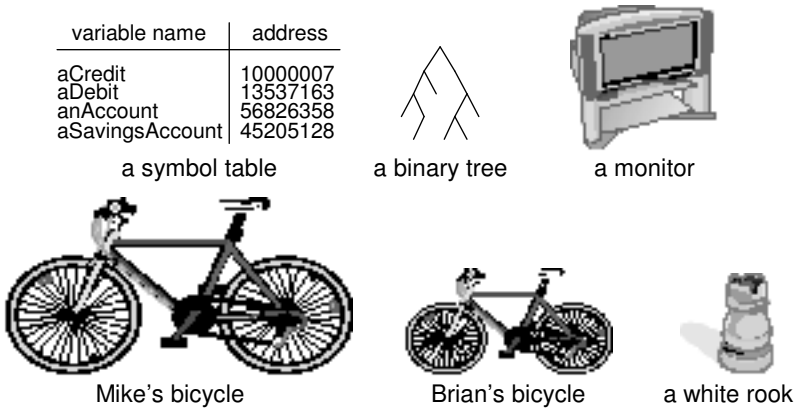

Figure 1: Objects: Objects lie at the heart of object –Oriented Technology

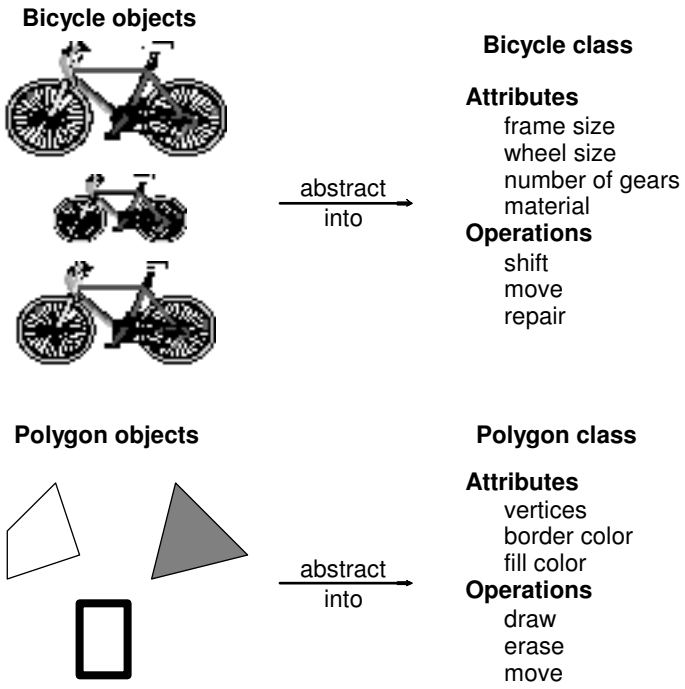

Figure2: Objects and classes. Each class describes a possibly infinite set of individual objects.

1.2. What is Object Oriented Development?

- Development refers to Software Life Cycle: analysis, design and implementation.
- Object oriented Development approach encourages software developers to work and think in terms of the application throughout the software life cycle
- OO Development is a conceptual process independent of programming language until the final stage.
- Greatest Benefit come from helping specifiers, developers and customers express abstract concepts clearly and communicate them to each other.
- It can serve as a medium for specification, analysis, documentation and interfacing as well as for programming

1.2.1 Modeling Concepts, not Implementation

.

- Emphasis here is on modeling and not pertains to any programming language.
- It is a fundamentally a way of thinking in the design perspective and not a programming technique.

1.2.2 Object Oriented Methodology:

The OO methodology has the following stages:

- **System conception:** Software development begins with business analyst or users conceiving an application and formulating tentative requirements.
- **Analysis:** The analysis model is a concise, precise abstraction of what the desired system must do, not how it will be done. The analysis model has two parts: the domain model and the application model.
- **System design.** The development team devises a high level strategy – the system architecture for solving the application problem. The system designer must decide what performance characteristics to optimize, chooses a strategy of attacking the problem and make tentative resource allocation.
- **Class design:** The class designer adds details to the analysis model in accordance with the system design strategy. The focus of the class design is the data structures and algorithms needed to implement each class.
- **Implementation:** Implementers translate the classes and relationships developed during class design into particular programming language, database or hardware. During implementation, it is important to follow good software engineering practice so that traceability to the design is apparent and so that the system remains flexible and extensible.

1.2.3 Three Models:

Three models to describe a system from different viewpoints:

- Class model: Describes the static structure of the objects in a system and their relationships (class diagrams).
- State Model: Describes the aspects of an object that change over time.(State Diagrams.)
- Interaction model: Describes how the objects in a system cooperate to achieve broader results. (Use cases, Sequence diagrams, activity diagrams.)

1.3 Object oriented Themes:

- Abstraction: let's focus on essential aspects of an application while ignoring details.
- Encapsulation :(Information Hiding) separates the external aspects of an object , that are accessible to other objects from internal implementation details.
- Combining data and Behavior.

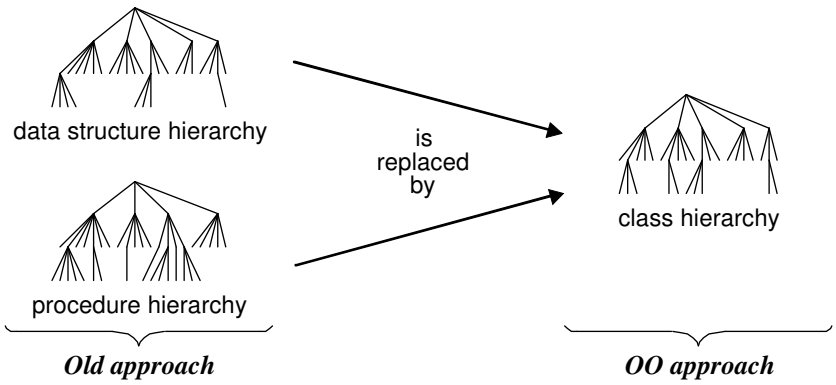

Figure3: OO vs Prior approach. An OO approach has one unified hierarchy for both data and behavior

- Sharing (reuse)
- Emphasis on the essence of an object:OO technology stresses what an object is, rather than how it is used.
- Synergy: Identity, classification, polymorphism, and inheritance characterize OO languages. Use all together.

1.4 Evidence for usefulness of OOD

- Applications at General Electric Research and Development Center.(1990)
- OO techniques for developing compilers, graphics, user interfaces, databases ,an OO language, etc.
- OO technology is a part of the computer science and software engineering mainstream.
- Important forums: (OOPSLA,ECOOP) Object Oriented Programming systems, Languages and applications. European Conference on OOP.
- IEEE Computer and Communications of the ACM.

1.5 OO Modeling history

- Work at GE Rand D led to OMT in 1991.
- Rumbaugh, Grady Booch on unifying the OMT and Booch Notaions in 1994.
- In 1996 the OMG issued a request for the proposals for a standard OO modeling notation.
- 1997 UML was accepted by OMG as a standard.
- In 2001 OMG released UML 1.Added features and released UML 2.)in 2004. www.omg.org

TEXT BOOKS:

1. **Object-Oriented Modeling and Design with UML–**Michael Blaha, James Rumbaugh , 2nd Edition, Pearson Education, 2005.
2. **Pattern-Oriented Software Architecture A System of Patterns, Volume 1 –** Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal John Wiley and Sons, 2006.

Advanced state modelling

4.0 Enumeraation

A data type is a description of values. Data types include numbers, strings and so on.

An enumeration is a data type that has a finite set of values. The advantage is to pick possible value and must restrict data to the legitimate values.

Some more examples

Weekdays = { Monday, Tuesday, … Sunday}

Months = { January, February , … December}

In UML , an enumeration is a data type. Enumeration can be declared by listing the keyword enumeration in guillemets (<< >>) above the enumerations name in the top section of a box.

4.1 Multiplcity

Multiplicity specifies the number of instances of one class that may relate to a single instance of an associated class. Multiplicity for an attribute specifies the number of possible values for each instantiation of an attribute. Mandatory single value [1] . Optional single value [0..1] and Many [*].

The scope indicates if a feature applies to an object or a class.

Visibility refers to the ability of a method to reference a feature from another class and has to possible values of public, protected and package.

Any method can freely access public features

Only methods of the containing class and its descendants via inheritance can access protected features

Only methods of the containing class can access private features

Methods of classes defined in the same package as the target class can access package features.

The UML denotes visibility with a prefix.

The character “+” precedes public features.

The character “#” precedes protected features.

The character “-” precedes private features.

The character “~” precedes package features.

There are few issues to consider when choosing visibility

Comprehension – Understanding all public features , the capabilities of a class.

Extensibility – Many classes can depend on public methods , so it can be highly disruptive to change their signature.

Context – Private, protected and package methods may rely on precondition or state information created by other methods in the class.

4.2 Association ends

Association end refers to the end of the association.

A binary association has two ends, a ternary association end has three and so on.

The features of an association are

- Association end name
- Multiplicity
- Ordering
- Bags and sequences
- Qualification

4.3 N-ary Associations

N-ary association means associations among three or more classes. You should try to avoid n-ary associations. Normally it is decomposed into binary associations with possible qualifiers and attributes.

4.4 Aggregation

Aggregation is a strong form of association in which an aggregate object is made of constituent parts. An aggregation as relating an assembly class to one constituent part class. An assembly with many kinds of constituent parts corresponds to many aggregations.

We know Aggregation is a special form of association. If two objects are tightly bound by a part-whole relationship, it is an aggregation. Aggregation is drawn like association, accept a small diamond indicates the assembly end.

The UML has two forms of part-whole relationships. A general form called aggregation and a more restrictive form called composition.

Composition is a form of aggregation with two additional constraints. A constituent part can belong to at most one assembly. Once a constituent part has been assigned an assembly, it has a coincident lifetime with the assembly.

4.5 propagation of operation

Propagation (Triggering) is the automatic application of an operation to a network of objects when the operation is applied to some starting object.

4.6 Abstract classes

An abstract class is a class that has no direct instances but whose descendant classes have direct instances. A concrete class is a class in which it can have direct instances.

4.7 Multiple Inheritance

Multiple Inheritance permits a class to have more than one superclass and to inherit features from all parents. The advantage is greater power to specify classes and increased reuse. The disadvantage is loss of conceptual and implementation simplicity. The most common form of multiple inheritance is from sets of disjoint classes. Multiple inheritance can also occur with overlapping classes.

4.8 Multiple classification

An instance of a class is inherently an instance of all ancestors of the class. For example an instructor could be both faculty and student. There is no class represents this. This is a simple example of multiple classification. UML permits multiple classification. But object oriented languages handles poorly. Dealing with a lack of multiple inheritance is really an implementation issue. Two approaches make use of delegation, which is an implementation mechanism by which an object forwards an operation to another object for execution.

4.9 Meta data

Metadata is data that describes other data.

4.10 Constraints

A constraint is a Boolean condition involving model elements, such as objects, classes, attributes, links, associations and generalization sets. A constraint restricts the values that elements can assume.

4.11 Derived data

A derived element is a function of one or more elements, which in turn may be derived.

4.12 packages

A package is a group of elements (classes, associations, generalizations and lesser packages) with a common theme.

Modeling as a Design Technique

2.0 Definition:

A model is an abstraction before building any system a prototype may be developed. The main purpose of model is for understanding of the system.

2.1 Introduction for modeling

Abstraction plays very important role in handling complexity. Abstraction means hiding detailed information and only important information is conveyed to the user. Engineers, artists build models for their designs. One can see even in the computer system. Operating system helps the user to use computer in a convenient and efficient way without knowing much about software and hardware present in the computer system.

2.2 Modeling

Designer build different kinds of models for various purposes before constructing things. For example car , airplane, pencil sketches for oil painting, blueprints of machine parts, Plan for house construction etc., Models serve many purposes

Testing a physical entity before building it

Engineers test scale models for airplanes, cars and boats in wind tunnels and water tanks to improve their dynamics. Also the use of simulation (Computer model) makes cheaper than building a complete system and correct if there is any flaw in it.

Communication with customers

Architects and product designers develop model to show their customers. Trial demonstration can be arranged.

Visualization

Storyboard of movies, television shows and advertisements show how their ideas flow.

Reduction of complexity

The human can understand if it is expressed in simple means. The model helps to bring complex things into simple.

2.3 Abstraction

Abstraction is process of masking unimportant details. For example the user see the automobile vehicle in a different way compared to a mechanic. All abstractions are incomplete and inaccurate. A good model essentially captures the important aspects of a problem and omits the others.

2.4 The three models

To capture the important aspects of the system in different viewpoints we use three models namely class model , state model and interaction model

The class model represents the static , structural , data aspects of a system.

The state model represents the temporal, behavioral , control aspects of system.

The interaction model represents the collaboration of individual objects of a system.

For example in a Software system, data structures part represents the class model , sequence operations in time represents the state model and communication among objects represents interaction model.

The three kinds of model separate a system into distinct views. The different models are not completely independent. Combination of all model results in better understanding of the system.

2.5 Class model

The class model describes the structure of objects in a system – their identity, their relationships to other objects , their attributes and their operations. The class model provides context for the state and interaction model.

Class diagrams representation used in the class model. Classes define the attribute values carried by each object and the operations that each object performs or undergoes.

2.6 State model

The state model describes those aspects of objects concerned with time and the sequencing of operations , events that mark changes, states that define the context for events and the organization of events and states.

The state model represents control i.e., the aspect of a system that describes the sequences of operations that occur, without regard for what the operations do, what they operate on, on how they are implemented.

A state diagram represents the state model. Each state diagram show the state and event sequences permitted in a system for one class of objects.

2.7 Interaction Model

The interaction model describes interaction between objects – how individual objects collaborate to achieve the behavior of the system as a whole.

Use cases, sequence diagrams and activity diagrams document the interaction model. In use case it shows the interaction between the system and outside actors. Sequence diagram show the objects that interact and the time sequence of their interactions. Activity diagrams show the flow of control among the processing steps of a computation.

state modelling

3.0 State modeling

The state model consists of multiple state diagrams, one for each class with temporal behavior.

3.1Events

An event is an occurrence at a point in time, such as user depresses left button to know various options. Another example Rail departs from Mysore to Bangalore.

Events corresponds to verbs in the past tense or to the onset of some condition. The time in which an event occurs is an implicit attribute of the event.

One event may logically precede or follow another or the two events may be unrelated. For example two flight can arrive at two different airports. Two events are casually unrelated are said to be concurrent , they have no effect on each other.

Events include error conditions and also normal occurrences. Example timeout events , paper jammed in printer, transaction aborted etc.,

There are several kinds of events

- Signal event
- Change event
- Time event

A signal is an explicit one-way transmission of information from one object to another. A signal event is the event of sending or receiving a signal. Here we are more concerned about the receipt of a signal, because it causes effects in the receiving object.

A signal is a message between objects while a signal event is an occurrence in time.

Every signal transmission is a unique occurrence, but we group them into signal classes and give each signal class a name to indicate common structure and behavior. Let us consider an example Kingfisher flight 123 departs from Mysore on March 23, 2010 is an instance of signal class Flight-Departure. Some signals are simple occurrences, but most signal classes have attributes indicating the values they convey. The UML notation is the keyword signal in guillements(<< >>) above the signal class in the top section of a box. The second section lists the signal attributes.

A change event is an event that is caused by the satisfaction of a Boolean expression.

Event of the expression is continually tested – whenever the expression changes from false to true , the event occurs. The UML notation for change event is the keyword when followed by a parenthesized Boolean expression.

A time event is an event caused by the occurrence of an absolute time or the elapse of a time interval. The UML notation for an absolute time is the keyword when followed by a parenthesized expression involving time. The notation for a time interval is the keyword after followed by a parenthesized expression that evaluates to a time duration.

3.2 State

A state is an abstraction of the values and links of an object. Set of values and links are grouped together into a state according to the grossbehavior of objects.

UML notation for a state – a rounded box, containing an optional state name.

The objects in a class have finite number of possible states. Each object can only be in one state at a time. Objects pass through one or more states during their lifetime.

A state specifies the response of an object to input events. All events are ignored in a state, except those for which behavior is explicitly prescribed. For example, if a digit is dialed in state Dial tone, the phone line drops the dial tone and enters state Dialing.

- Events Vs States

Events represent points in time. State represent intervals of time. A state corresponds to the interval between two events received by an object.

Both events and states depend on the level of abstraction. A travel agent planning an itinerary would treat each segment of a journey as a single event, an air traffic control system would break each flight into many geographical legs.

3.3 Transitions and conditions

A transition is an instantaneous change from one state to another. For example, when a phone is answered , the phone line transitions from the Ringing state to the Connected state. The choice of next state depends on both the original state and the event received. A guard condition is a Boolean expression that must be true in order for a transition to occur. A guard condition is checked only once, at the time the event occurs and the transition fires, if the condition is true.

3.4 State diagrams

A state diagram is a graph whose nodes are states and whose directed arcs are transitions between states. A state diagram specifies the state sequences caused by event s. The state model consists of multiple state diagrams, one state diagram for each class with important temporal behavior. Sequences. State names must be unique within the scope of a state diagram.

3.5 One shot state diagram

The state model consists of multiple state diagrams, one state diagram for each class with important temporal behavior.

Object Oriented Modelling and Design Notes eBook

Publisher : VTU eLearning

Author :

Type the URL : http://www.kopykitab.com/product/1840

Get this eBook