Chapter 2

Underpinnings of Requirements Analysis

Copyright © 2000 by Addison Wesley
Version 1.0

Topics

- Fundamentals of Object Technology
  - Instance object
  - Class
  - Association
  - Aggregation and Composition
  - Generalization
  - Class Object
- Guided Tutorial in Analysis Modeling
  (ref. separate set of slides)
- Problem Statements for Case Studies
**Fundamentals of OT**

- **Object has**
  - State
  - Behavior
  - Identity

- **Objects and natural systems**

---

**Instance object**

- **Class**
- **Instance object**
- **Class object**

```
c1: Course
    course_number = COMP227
    course_name = Requirements Analysis and System Design
```
How objects collaborate?

- Order
  - shipOrder()
- Shipment
  - subtractProducts()
- Stock
  - analyzeStockLevels()
- Purchase
  - reorderProducts()

How objects identify each other?

- OID
- OID links
- Object longevity
  - Persistent object
  - Transient object
- Object communication via
  - Persistent OIDs
  - Transient OIDs
Persistent link implementation

```
c1: Course

  course_number = COMP227
  course_name = Requirements Analysis and System Design
  teacher: identity = Ref@#$%
```

Persistent links in UML

```
c1: Course

  course_number = COMP227
  course_name = Requirements Analysis and System Design

  c2: Course
    teacher

  t1: Teacher
    course
    teacher
```
Transient link

- How does an object know the OID of another object if there is no persistent link?
  - Search on the database
  - A “map” object
  - Creating a new object

- Pointer swizzling

Class

<table>
<thead>
<tr>
<th>Class name</th>
<th>Attributes</th>
<th>Operations()</th>
</tr>
</thead>
</table>

(c) Addison Wesley
**Attribute**

<table>
<thead>
<tr>
<th>Course</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>course_number : String</td>
<td>order_number : Integer</td>
</tr>
<tr>
<td>course_name : String</td>
<td>order_date : Date</td>
</tr>
<tr>
<td></td>
<td>order_value : Currency</td>
</tr>
</tbody>
</table>

**Attribute type designating a class**

<table>
<thead>
<tr>
<th>Order</th>
<th>Shipmen t</th>
</tr>
</thead>
<tbody>
<tr>
<td>order_number : Integer</td>
<td>the_order</td>
</tr>
<tr>
<td>order_date : Date</td>
<td>the_shipment</td>
</tr>
<tr>
<td>order_value : Currency</td>
<td>shipOrder()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Order</th>
<th>Shipmen t</th>
</tr>
</thead>
<tbody>
<tr>
<td>order_number : Integer</td>
<td>the_order : Order</td>
</tr>
<tr>
<td>order_date : Date</td>
<td>the_shipment : Shipmen t</td>
</tr>
<tr>
<td>order_value : Currency</td>
<td>shipOrder()</td>
</tr>
</tbody>
</table>
Attribute visibility

```
Purchase
- purchase_number : String
- purchase_date : Date
- purchase_value : Currency

reorderProducts()
```

Operation

```
Order

Stock
- subtractProducts()
- analyzeStockLevels()

Shipment
- shipOrder()

Purchase
- reorderProducts()
```
**Association**

```
Order
- order_number : Integer
- order_date : Date
- order_value : Currency

+ the_order *
```

```
Shipment
- shipment_id : String
- shipment_date : Date
- carrier : String

+ shipOrder()
```

```
OrdShip

+ the_shipment *
```

**Association degree**

- **Binary**
- **Unary (singular)**
- **Ternary**

```
Employee 0..1

+ can_be_manager_of 0..*
```

```
+ can_be_managed_by
```
Association multiplicity

- 0..0
- 0..1
- 0..*
- 1..1
- 1..*
- *

Teacher

```
+----------------+-------+
| taught_by      | 1..*  |
+----------------+-------+

is_managed_by

```

```
Teacher

<table>
<thead>
<tr>
<th>is_in_charge_of</th>
<th>0..*</th>
</tr>
</thead>
</table>
```

CourseOffering

Association link and extent

- Link – association instance
- Extent – set of association instances

Order 1

```
+----------------+-------+
| Link 1         | Link 2 |
| with           |       |
| 3 references   |       |
+----------------+-------+
```

Order 2

```
+----------------+-------+
| Link 3         |      |
+----------------+-------+

Shipment 1

```
+----------------+-------+
| Link 4         |      |
+----------------+-------+
```

Shipment 2

```
+----------------+-------+
| Link 5         |      |
+----------------+-------+
```

Shipment 2
**Association class**

```
<table>
<thead>
<tr>
<th>Assessment</th>
</tr>
</thead>
<tbody>
<tr>
<td>mark : List(Number)</td>
</tr>
<tr>
<td>total_mark : Number</td>
</tr>
<tr>
<td>grade : Byte</td>
</tr>
</tbody>
</table>
```

```
CourseOffering * Student *
```

**Composition and aggregation**

- **Composition** – aggregation by value
- **Aggregation** – aggregation by reference
- **Properties:**
  - Transitivity
  - Asymmetry
  - Existence dependency

```
Book

<table>
<thead>
<tr>
<th>is_part_of</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>has</th>
</tr>
</thead>
</table>

Chapter

<table>
<thead>
<tr>
<th>*</th>
</tr>
</thead>
</table>

Section

<table>
<thead>
<tr>
<th>*</th>
</tr>
</thead>
</table>

Crate

<table>
<thead>
<tr>
<th>*</th>
</tr>
</thead>
</table>

BeerBottle

```
Generalization

- Inheritance
- Reuse

Polymorphism

The same signature
(operation name and the number and type of arguments)
Multiple inheritance

Person
- full_name : String
- date_of_birth : Date
- age()

Teacher

Student

PostgraduateStudent

Tutor

Multiple classification

- **Multiple inheritance**
  - A class may have many superclasses, but a single class must be defined for each object

- **Multiple classification**
  - An object is simultaneously the instance of two or more classes

- The problem arises if Person is specialized in few orthogonal hierarchies
  - Person can be Employee or Student, Male or Female, Child or Adult, etc.

- Without multiple classification
  - need to define classes for each legal combination between the orthogonal hierarchies
  - ChildFemaleStudent etc.
Dynamic classification

- An object does not only belong to multiple classes but it can gain or lose classes over its lifetime.
- A Person object can be just an employee one day and a manager (and employee) another day.
- In most current object-oriented programming environments, an object cannot change its class after it has been instantiated (created).

Abstract class

- Parent class that will not have direct instance objects.
- Abstract class cannot instantiate objects because it has at least one abstract operation.

Diagram:

```
    Video
     ↓
  <<abstract>> rentalCharge()
    ↓
VideoTape  VideoDisk
```

(c) Addison Wesley Chapter 2 25

(c) Addison Wesley Chapter 2 26
Class object

- **Object with**
  - Class-scope attributes and/or
  - Class-scope operations

<table>
<thead>
<tr>
<th>Student</th>
</tr>
</thead>
<tbody>
<tr>
<td>student_id : String</td>
</tr>
<tr>
<td>student_name : PersonName</td>
</tr>
<tr>
<td>max_courses_per_semester : Integer</td>
</tr>
<tr>
<td>&lt;&lt;global&gt;&gt; averageStudentAge() : Real</td>
</tr>
</tbody>
</table>

Statements for case studies

- **University Enrolment**
- **Video Store**
- **Contact Management**
- **Telemarketing**
University Enrolment

- **The university offers**
  - Undergraduate and postgraduate degrees
  - To full-time and part-time students

- **The university structure**
  - Divisions containing departments
  - Single division administers each degree
  - Degree may include courses from other divisions

- **University enrolment system**
  - Individually tailored programs of study
  - Prerequisite courses
  - Compulsory courses
  - Restrictions
    - Timetable clashes
    - Maximum class sizes, etc.

University Enrolment (cont)

- The system is required to
  - Assist in pre-enrolment activities
  - Handle the enrolment procedures

- **Pre-enrolment activities**
  - Mail-outs of
    - Last semester's examination grades to students
    - Enrolment instructions

- **During enrolment**
  - Accept students’ proposed programs of study
  - Validate for prerequisites, timetable clashes, class sizes, special approvals, etc.

- **Resolutions to some of the problems may require consultation with academic advisers or academics in charge of course offerings**
Video Store

- The **video store**
  - Rentals of video tapes and disks to customers
  - All video tapes and disks bar-coded
  - Customer membership also be bar-coded.
- Existing customers can place reservations on videos to be collected at specific date
- Answering customer enquiries, including enquiries about movies that the video store does not stock (but may order on request)

Contact Management

- The **market research company** with established customer base of organizations that buy market analysis reports
- The company is constantly on the search for new customers
- **Contact management** system
  - Prospective customers
  - Actual customers
  - Past customers
- The new contact management system to be developed internally and be available to all employees in the company, but with varying levels of access
  - Employees of Customer Services Department will take the ownership of the system
- The system to permit flexible scheduling and re-scheduling of contact-related activities so that the employees can successfully collaborate to win new customers and foster existing relationships
Telemarketing

- **The charitable society** sells lottery tickets to raise funds
  - **Campaigns** to support currently important charitable causes
  - Past contributors (**supporters**) targeted through telemarketing and/or direct mail-outs

- **Rewards (special bonus campaigns)**
  - For bulk buying
  - For attracting new contributors

- The society does not randomly target potential supporters by using telephone directories or similar means

---

Telemarketing (cont)

- **Telemarketing application**
  - To support up to fifty telemarketers working simultaneously
  - To schedule the phone calls according to pre-specified priorities and other known constraints
  - To dial up the scheduled phone calls
  - To re-schedule unsuccessful connections
  - To arrange other telephone callbacks to supporters
  - To record the conversation outcomes, including ticket orders and any changes to supporter records
Summary

- Each object has a state, behavior and identity
- Class defines attributes and operations
- There are three kinds of relationships – association, aggregation, generalization
- Generalization provides the basis for polymorphism and inheritance
- Multiple inheritance is likely to be supported
- Multiple and dynamic classification is still not supported commercially
- Abstract classes are important in modeling
- There are instance objects and class objects
- The OnLine Shopping guided tutorial (separate Lecture Notes)
- Four case studies