Lecture 6, Part 2: Modelling Objects

Jennifer Campbell
CSC340 - Winter 2007

Analysis vs. Design

Reminder: we are modelling this and this … … but not this

Application Domain
D - domain properties
R - requirements
CSC340 University of Toronto 2

Machine Domain
S - specifications
C - computers
P - programs

Analysis models:
• represent people, physical things and concepts important to the application domain
• show connections and interactions among these people, things and relevant concepts
• allow us to check whether the functions we will include in the specification will satisfy the requirements

CSC340 University of Toronto 3

Object Oriented Analysis (OOA)

• OOA grew out of object oriented design
  – Applied to modelling the application domain rather than the program
  – Model the requirements in terms of objects and the services they provide
• OO is (claimed to be) more ‘natural’
  – As a system evolves, the functions change more often than the objects on which they operate
  – a model based on objects (rather than functions) will be more stable over time
  – hence the claim that object-oriented designs are more maintainable

CSC340 University of Toronto 4

Classes

• A class describes a group of objects with
  – similar properties (attributes),
  – common behaviour (operations),
  – common relationships to other objects,
  – and common meaning (“semantics”).
• Example:
  – Employee: has a name, employee number and department; an employee is hired, and fired; an employee works on one or more projects

CSC340 University of Toronto 5

Objects vs. Classes

• The instances of a class are called objects.

<table>
<thead>
<tr>
<th>Fred_Bloggs:Employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>name: Fred Bloggs</td>
</tr>
<tr>
<td>employee number: 234609234</td>
</tr>
<tr>
<td>department: Marketing</td>
</tr>
</tbody>
</table>

– Two different objects may have identical attribute values (like two people with identical name and address)
• Objects have associations with other objects
  – E.g. Fred_Bloggs:employee is associated with the KillerApp:project object
  – But we will capture these relationships at the class level (why?)
• Note: Make sure attributes are associated with the right class
  • E.g. you don’t want both managerName and manager# as attributes of Project! (…Why??)

CSC340 University of Toronto 6

UML Class Diagrams

• UML Class Diagrams show classes and their relationships
• Relationships: connections between classes
  – Objects do not exist in isolation from one another
  – Types of UML relationships:
    • Association
    • Aggregation and Composition
    • Generalization
    • Dependency
    • Realization {not useful for analysis}
Class Associations

- **Class Associations**
  - **Multiplicity**
  - **A Client has exactly one StaffMember as a contact person**
  - **Name of the association**
  - **StaffMember**
    - name
    - id number
    - start date
    - contact person

Association Multiplicity

- **Multiplicity**: the minimum and maximum times an object can be associated with the related object
- **Examples**:
  - Optional (0 or 1) 0..1
  - Exactly one 1 = 1..1
  - Zero or more 0..*
  - One or more 1..*
  - A range of values 1..6
  - A set of ranges 1..3, 7..10, 15, 19..*

Exercise: Multiplicities

- **Patient**
  - insurance carrier
  - book appointment
  - explain symptoms
- **Appointment**
  - time
  - date
  - reason
  - cancel without notice
- **Doctor**
  - primary office
  - specialization
  - assess patient
- **Does a doctor need to attend appointments?**
- **Can a doctor attend multiple appointments?**
- **Does a patient have to schedule an appointment?**
- **Can a patient schedule more than one appointment?**

Association Classes

- **Sometimes the association is itself a class**
  - …because we need to retain information about the association
  - …and that information doesn’t naturally live in the classes at the ends of the association
  - E.g. a "title" is an object that represents information about the relationship between an owner and her car

Aggregation

- **The “is part of” or “whole-part” relationship**
- **An employee is part of a team.**

Composition

- **Strong form of aggregation that implies ownership**:
  - if the whole is removed from the model, so is the part
  - the whole is responsible for the disposition of its parts

Note: The multiplicity should be 1 when the association is composition (i.e., a car should always have (at least) one engine).
Generalization

Subclasses inherit attributes, associations, & operations from the superclass

- Person
  - name
  - SIN

- Professor
  - employee id
  - department
  - salary
  - teach course

- Student
  - student id
  - major
  - add course
drop course

- Address
  - street
city
province
postal code

lives at

0..* 1

More on Generalization

- Usefulness
  - Can easily add new subclasses if the organization changes

- Look for generalizations in two ways:
  - Top Down
    - Subdivide an existing class
    - Or you have an association that expresses a “kind of” relationship
    - E.g. “Most of our work is on advertising for the press, that’s newspapers and magazines.”
  - Bottom Up
    - You notice similarities between classes you have identified
    - E.g. “We have books and we have CDs in the collection, but they are all filed using the Dewey system, and they can all be lent out and reserved”

More on Generalization [2]

- Don’t generalize just for the sake of it
  - Be sure that everything about the superclass applies to the subclasses
  - Be sure that the superclass is useful as a class in its own right
  - Don’t add subclasses or superclasses that are not relevant to your analysis

Finding Classes

- Look for nouns and noun phrases in stakeholders’ descriptions of the problem
  - include in the model if they explain the nature or structure of information in the application.

- It’s better to include many candidate classes at first
  - You can always eliminate them later if they turn out not to be useful
  - Explicitly deciding to discard classes is better than just not thinking about them

Possible Classes

- External Entities
  - …that interact with the system being modeled
    - E.g. people, devices, other systems

- Things
  - …that are part of the domain being modeled
    - E.g. reports, displays, signals, etc.

- Occurrences or Events
  - …that occur in the context of the system
    - E.g. transfer of resources, a control action, etc.

Possible Classes [2]

- Roles
  - played by people who interact with the system

- Organizational Units
  - that are relevant to the application
    - E.g. division, group, team, etc.

- Places
  - …that establish the context of the problem being modeled
    - E.g. manufacturing floor, loading dock, etc.

- Structures
  - that define a class or assembly of objects
    - E.g. sensors, four-wheeled vehicles, computers, etc.
Selecting Classes

• Discard classes for concepts which:
  – Are beyond the scope of the analysis
  – Refer to the system as a whole
  – Duplicate other classes
  – Are too vague or too specific
    • e.g. have too many or too few instances

• Include external entities as classes if they:
  – Produce or consume information essential to
    the system

Coad & Yourdon’s Criteria for Selecting Classes

• Retained Information: Will the system need to remember information about this class of objects?
• Needed Services: Do objects in this class have identifiable operations that change the values of their attributes?
• Multiple Attributes: Does the class have multiple attributes?
• Common Attributes: Does the class have attributes that are shared with all instances of its objects?
• Common Operations: Does the class have operations that are shared with all instances of its objects?

Exercise: Online Order System

A Customer places their Order on the web. The Customer has an address, and provides a form of payment. The Order includes the quantities of each Product that they want to have delivered. In order to plan deliveries and allocate delivery vans, Groceries-R-Us must determine the volume of orders being delivered, so in addition to the name, cost price and sale price of each Product, they also need to know the cubic volume required for a single item of that Product.

Modelling Notations

• So far we’ve seen:
  – UML Activity Diagrams
    • Show the sequence of steps that make up a business process (workflow).
  – UML Class Diagrams
    • static domain model
    • structure of the domain

References

