Session Outline
1. Visualizing Concepts
2. Associations between Concepts
3. Attributes of Concepts

*Modeling the real world – not software objects!*

*“Understand the elements of the problem space before you attempt to design the solution”*

---

**Partial Domain Model**

*“visual dictionary”: easier for the brain to process than text*

**Domain Concept vs. Software Class**

In domain modeling, we omit:
- software artifacts like windows, databases, ...
- responsibilities or methods

[Larman, 2002]

---

**Symbol, Intension, Extension**

Chosen name for the conceptual class

Design View

Textual description / definition

Run-Time View

Set of all instances of the concept

[Larman, 2002]

---

**Choices in Class Naming**

similar concepts with different names

Post or?

Register

Sale

[Larman, 2002]
Identifying Conceptual Classes

- Use a conceptual class category list:
  - Physical or tangible objects
    - Register, Airplane
  - Specifications, descriptions of things
    - ProductSpecification, FlightDescription
  - Places
    - Store, Airport
  - Transactions
    - Sale, Payment, Reservation
  - etc. (see Table 10.1 for a fuller list)

Identifying Classes [2]

- Noun Phrase Identification [Abbot 83]
  - Analyze textual description of the domain
  - Identify nouns and noun phrases
    - (indicate candidate classes or attributes)
  - Caveats:
    - Automatic mapping isn’t possible
    - Textual descriptions are ambiguous!
      - (different words may refer to the same class)

Part-of-speech Analysis \[\text{Abbot 1983}\]

<table>
<thead>
<tr>
<th>Part of speech</th>
<th>Model component</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proper noun</td>
<td>object</td>
<td>Jim Smith</td>
</tr>
<tr>
<td>Improper noun</td>
<td>class</td>
<td>man</td>
</tr>
<tr>
<td>Doing verb</td>
<td>method</td>
<td>buy, recommend</td>
</tr>
<tr>
<td>being verb</td>
<td>inheritance</td>
<td>is-a (kind-of)</td>
</tr>
<tr>
<td>having verb</td>
<td>aggregation</td>
<td>has an</td>
</tr>
<tr>
<td>modal verb</td>
<td>constraint</td>
<td>must be</td>
</tr>
<tr>
<td>adjective</td>
<td>attribute</td>
<td>3 years old</td>
</tr>
<tr>
<td>transitive verb</td>
<td>method</td>
<td>enter</td>
</tr>
<tr>
<td>intransitive verb</td>
<td>method (event)</td>
<td>depends on</td>
</tr>
</tbody>
</table>

Surf Report: Language Analysis

SurfReport will email users up to 4 messages per day at user-specified times with information about the surfing conditions in their area. The user can specify which location/s he/she would like reported, and what information (tide levels, wind, waves, etc.) he/she would like for each location.

Noun (class)
Verb (operation)
Adjective (Attribute, Constraint)

Creating a Domain Model

1. List the candidate conceptual classes using the two techniques, based on the requirements
2. Draw them in a domain model
3. Add associations necessary to record relationships
4. Add attributes to meet information requirements

The “Mapmaker” Strategy

- “Use existing names in the territory” (use the domain vocabulary)
- “Exclude irrelevant features” (not relevant to the business process)
- “Don’t add things that aren’t there” (outside the domain)
Description Classes

What is the rationale for representing the description or specification separately from the item?

ProductSpecification

<table>
<thead>
<tr>
<th>Description</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>price</td>
</tr>
<tr>
<td>serial number</td>
<td>entered</td>
</tr>
</tbody>
</table>

[Description Classes]

Another Example...

What is the rationale for representing the description separately from the flight?

Flight

<table>
<thead>
<tr>
<th>Describe</th>
<th>FlightDescription</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>time</td>
</tr>
<tr>
<td>number</td>
<td>serial number</td>
</tr>
</tbody>
</table>

[Another Example...]

Domain Model vs. Design Model

A Class is a Class is a Class...

- Conceptual Class (UP Domain Model) real-world concept or thing
- Design Class (UP Design Model) generally a software class
- Software Class (Implementation View) a software component specification
- Implementation Class (Implementation) e.g. a class written in Java

Domain Model vs. Design Model

<table>
<thead>
<tr>
<th>Payment</th>
<th>Sale</th>
</tr>
</thead>
<tbody>
<tr>
<td>amount</td>
<td>date</td>
</tr>
<tr>
<td>1: pay for 1</td>
<td>data time</td>
</tr>
</tbody>
</table>

UP Domain Model

Raw UML class diagram notation used in an essential model visualizing real-world concepts.

UP Design Model

New UML class diagram notation used in a specification model virtualizing software components.

[Domain Model vs. Design Model]

Closing the Representational Gap

NextGen Concept Classes

Register | Item | Store | Sale |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sales Lineman</td>
<td>Cashier</td>
<td>Customer</td>
<td>Manager</td>
</tr>
</tbody>
</table>

Payment | Product Catalog | Product Specification

[Closing the Representational Gap]

[NextGen Concept Classes]
Associations

Influence of Domain Model

Associations in UML

Finding Associations
- Use the common associations list
  - A is a physical part of B
  - A is a logical part of B
  - A uses or submanages B
  - A communicates with B
  - etc. (see p. 156 for a fuller list; NextGen example on page 162)

Multiplicity

Multiplicity Values
Data Types as Attributes

Attributes Aren’t Foreign Keys

Modeling Quantities

Domain Model with Attributes

Quantities and Derived Attributes

Partial Domain Model

[Larman, 2002]
Questions and Discussion