Conceptual Modeling

Modeling the Problem Domain

- Decompose problem space into comprehensible concepts.
- Clarify the terminology or vocabulary of the problem domain.
- Includes concepts, associations between concepts, and attributes of concepts.
- Does not include responsibilities, methods, or software artifacts such as window or database.

Class diagrams (POS Example)
Concepts

- Symbol
  - Words or images representing a concept
  - Sale

- Intension
  - The definition of a concept
  - “The event of a purchase transaction...”

- Extension
  - The set of instances of the concept
  - The set of all sales

Conceptual Modeling Process

1. Identify candidate concepts
2. Remove poor candidates
3. Add associations
4. Add attributes
5. Add inheritance
6. Refine

Identifying Concepts (1)

- Use existing names as they are used in the problem domain.
  - In the library domain there are “Borrowers” or “Patrons”, not “Customers”

- Exclude irrelevant concepts
  - “Shopping bag” has no role in the requirements

- Don’t invent concepts that are not already there
Identifying Concepts (2)

• Extract nouns and noun phrases from use case descriptions.
• Consider a list of concept categories:

Physical objects (POST)
- Designs, descriptions (Product Spec.)
- Places (Store)
- Transactions (Sale, Payment)
- People’s roles (Cashier)
- Containers (Stor, Bin)
- Contained things (Item)
- External systems (Credit bureau)
- Abstract concepts (Hunger)

Organizations (Sales Dept.)
- Events (Sale, Robbery)
- Policies (Refund policy)
- Catalogs (Product catalog)
- Records/Contracts (Receipt)
- Financial Items (Credit line)
- Manuals/Books (Employee manual)

Concept Quality

Good Concepts
- POST
- Sale
- Line item
- Customer
- Cashier
- Account
- Store
- Product

Poor Concepts
- System
- Security Provision
- Sales price
- User
- Cash

Receipt?

• Receipt is a report of a sale.
• Reports are generally not useful in conceptual models, since all their information is derived from other sources.

• Receipt confers the right to return an item to the store for a refund.
• If an item return use case is included, it might be useful to include receipts in the conceptual model.
Attributes and Concepts

• If we think of something in the real world as text or a number it is probably an attribute, otherwise it is a concept.

• Is destination an attribute of a flight, or a concept?

• When in doubt, use a separate concept.

When to use specifications

• When deleting the last instance of something will incorrectly result in the loss of information.

• When it reduces redundant information.

Adding Associations

• Associations should imply an enduring relationship which needs to be remembered.
  – Association between a sale and its line items.
  – No association between a sale and a sales manager.
  – No association between the POST and a customer’s credit card.
Common Associations

- Physical part (POST–Drawer)
- Logical part (LineItem–Sale)
- Physical containment (POST–Store)
- Logical containment (Description–Catalog)
- Description (ProductSpec–Item)
- Knows/Records (POST–Sale)
- Membership (Cashier–Store)
- Ownership (POST–Store)
- Subunit (Department–Store)
- Transaction (Payment–Customer, Payment–Sale)
- Communication (Customer–Cashier)

Association Guidelines (1)

- Focus on enduring relationships rather than transient events:
  - Concepts are more important than associations.
  - Too many associations may confuse the model.

Association Guidelines (2)

- Avoid redundant and derivable associations unless they improve domain understanding.
  - Can any of these associations be removed?
Aggregation and Composition

- *Part-of* associations.
- Composition is a strong form of aggregation where the parts may belong to only one whole, and whose existence depends on the whole.
- But what does this really mean?

```
Company  Division  Person
```

Multiplicity

<table>
<thead>
<tr>
<th>Multiplicity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>zero or more</td>
</tr>
<tr>
<td>1..*</td>
<td>one or more</td>
</tr>
<tr>
<td>1..10</td>
<td>one to ten</td>
</tr>
<tr>
<td>5</td>
<td>exactly five</td>
</tr>
<tr>
<td>3, 5, 10...*</td>
<td>three, five, or at least 10</td>
</tr>
</tbody>
</table>

Examples

```
Employee * supervision

Flight  Airline
        * Flies-to 0..1
        * Flies-from 1

Airline
```
Adding Attributes

- Attributes should be simple, pure data values.
  - Number, String, Date, Color, etc.
  - Not flight destination
- Relate concepts with associations, not attributes.
  - Don’t use attributes as “foreign keys”
  - Item does not have UPC attribute, it has an association to ProductSpec

Inheritance

- Share common attributes and associations.
- Bottom-up
  - Generalize common aspects of existing classes
- Top-down
  - Specialize existing classes

Key Points (1)

- Conceptual models are concerned with the problem domain, not software artifacts.
- UML class diagrams are useful for modeling concepts and associations.
- Don’t confuse attributes with concepts.
- Concepts are more important than associations and should receive more attention.
Key Points (2)

• Don’t include associations that are redundant or derivable from other associations unless they improve domain comprehension.

• 80% of modeling activities use only 20% of UML notation. Don’t feel compelled to use every feature (e.g. aggregation, composition).