Multiplicity

One person holds one credit card

- One object can be related to many objects through the same association

One person can hold zero or more credit cards
• One person can hold zero or more credit cards (0..*)
• Each card has zero or one holder (0..1)

<table>
<thead>
<tr>
<th>Person</th>
<th>0..1 Holds</th>
<th>0..* Credit-card</th>
</tr>
</thead>
<tbody>
<tr>
<td>name: String</td>
<td></td>
<td>card-number: integer</td>
</tr>
<tr>
<td>age: integer</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

`:JQPublic:Person` 
name= J. Q. Public 
age=35
Holds
Card789:Credit-Card 
card-number= 1233 4566 7899 2101
Card123:Credit-Card 
card-number= 1111 2222 3333 4444

`:DQPublic:Person` 
name= D. Q. Public 
age=32
Card456:Credit-Card 
card-number= 4444 5555 6666 7777
• One person can hold zero or more credit cards (0..*)
• Each card has one holder (no indication or 1)
• Each card has one or more authorized users (1..*)
• One person can be authorized to use zero or more cards

Explicit enumeration is also possible (2, 3, 2..5, etc.)
Link Attributes

- Associations can have properties the same way objects have properties.

<table>
<thead>
<tr>
<th>Person</th>
<th>0..* Works-for</th>
<th>Company</th>
</tr>
</thead>
<tbody>
<tr>
<td>name: String</td>
<td></td>
<td>name: String</td>
</tr>
<tr>
<td>age: integer</td>
<td></td>
<td>address: String</td>
</tr>
<tr>
<td>SSN: integer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>address: String</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

How to represent salary and job title?

Use a link attribute!
Folding Link Attributes

Person

| name: String |
| age: integer |
| SSN: integer |
| address: String |

Company

| name: String |
| address: String |

0..* Works-for

Why not this?

Salary and job title are properties of the job not the person

In this case, a link attribute is the only solution

... or not
Alternatives...

- **Employee**
  - name: String
  - age: integer
  - SSN: integer
  - address: String
  - salary: integer
  - job-title: String

- **Company**
  - name: String
  - address: String

- **Person**
  - name: String
  - age: integer
  - SSN: integer
  - address: String

- **Position**
  - salary: integer
  - job-title: String
  - filled: Boolean

- **Relationships**
  - Employee works for Company
  - Company has Position
  - Position is filled by Person

- **Cardinalities**
  - Employee works for Company: 0..*
  - Company has Position: 1
  - Position filled by Person: 0..1


- Attach names to the ends of an association to clarify its meaning
Aggregation

● A special association, the is-part-of association
  ▪ A sentence is part of a paragraph (a paragraph consists of sentences)
  ▪ A paragraph is part of a document (a document consists of paragraphs)
Aggregation (Cont.)

- Often used in parts explosion
Generalization and Inheritance

- The is-a association
  - Cards have many properties in common
  - Generalize the common properties to a separate class, the base-card
  - Let all cards inherit from this class, all cards is-a base-card (plus possibly something more)
**Association**

- Can you use the phrase is-part-of or is-made-of
- Are operations automatically applied to the parts (for example, move) - aggregation
- Not clear what it should be......
Aggregation Versus Inheritance

- Do not confuse the is-a relation (inheritance) with the is-part-of relation (aggregation)
- Use inheritance for special cases of a general concept
- Use aggregation for parts explosion
Recursive Aggregates

- A recursive aggregate contains (directly or indirectly) an instance of the same kind of aggregate
Object Modeling Summary

- **Classes**
  - Name
  - Attributes
  - Operations

- **Associations**
  - Roles
  - Link attributes

- **Aggregation**
- **Inheritance**
Object Modeling Approach

- Start with a problem statement
  - High-level requirements

- Define object model
  - Identify objects and classes
  - Prepare data dictionary
  - Identify associations and aggregations
  - Identify attributes of objects and links
  - Organize and simplify using inheritance
  - Iterate and refine the model
  - Group classes into modules
The Home Heating System

Fuel Valve

Hot Water

Fuel

Burner

Controller

Water Pump

Water Valve

Control Panel

90
80
70
60
50

Temp Sensor

Home

The Home Heating System
Home Heating Requirements

The purpose of the software for the Home Heating System is to control the heating system that heats the rooms of a house. The software shall maintain the temperature of each room within a specified range by controlling the heat flow to individual rooms.

- The software shall control the heat in each room
- The room shall be heated when the temperature is 2F below desired temp
- The room shall no longer be heated when the temperature is 2F above desired temp
- The flow of heat to each room shall be individually controlled by opening and closing its water valve
- The valve shall be open when the room needs heat and closed otherwise
- The user shall set the desired temperature on the thermostat
- The operator shall be able to turn the heating system on and off
- The furnace must not run when the system is off
Home Heating Requirements

The purpose of the software for the Home Heating System is to control the heating system that heats the rooms of a house. The software shall maintain the temperature of each room within a specified range by controlling the heat flow to individual rooms.

- When the furnace is not running and a room needs heat, the software shall turn the furnace on.
- To turn the furnace on the software shall follow these steps:
  - open the fuel valve
  - turn the burner on
- The software shall turn the furnace off when heat is no longer needed in any room.
- To turn the furnace off the software shall follow these steps:
  - close fuel valve
  - turn burner off
Identify Object Classes

Candidate Classes

Water Pump
Hot Water
Burner
Fuel Valve
Fuel

controller
operator
furnace
thermostat
heating system

Home Heating System
Fire Valve
range

house
room
Water Valve

software
temperature
heat
Temp Sensor
Control Panel

on-off switch
desired temp

fuel

heat
weather

desired temp
temperature

software

heat
weather

home
Eliminate Bad Classes

- Redundant classes
  - Classes that represent the same thing with different words

- Irrelevant classes
  - Classes we simply do not care about

- Vague classes
  - Classes with ill-defined boundaries
Eliminate Classes

Redundant
- heating system
- user

Irrelevant
- Fuel
- software
- Hot Water

Vague
- heat
- house
- heat flow
- home
- range
- desired temp
- temperature

Attributes

Operations
- None

Roles
- None

Implementation
- None

- Fuel Valve
- Water Pump
- thermostat
- on-off switch
- Burner
- furnace
- operator
- room
- Home Heating System
- Temp Sensor
- Water Valve
- Control Panel
Classes After Elimination

- Fuel Valve
- Water Pump
- Thermostat
- Operator
- Controller
- Furnace
- Room
- Home Heating System
- Burner
- Temp Sensor
- Water Valve
- on-off switch
- Control Panel
Defining the Classes

- **Attributes**
  - Things that describe individual objects

- **Operations**
  - Sequences of actions are often mistaken for classes

- **Roles**
  - The name of a class should reflect what it is, not the role it plays

- **Implementation details**
  - Save that for implementation
Prepare Data Dictionary

- Water Tank
  - The storage tank containing the water that circulates in the system.

- Pump-1
  - The pump pumping water from the Water Tank to the radiators in the rooms.
Possible Associations

- Not much information from the prose requirements
- A lot of information from the system design

- A room consists of a thermometer and a radiator
- A radiator consists of a valve and a radiator element
- The home heating system consists of a furnace, rooms, a water pump, a control panel, and a controller
- The furnace consists of a fuel pump and a burner
- The control panel consists of an on-off switch and a thermostat
- The controller controls the fuel pump
- The controller controls the burner
- The controller controls the water pump
- The controller monitors the temperature in each room
- The controller opens and closes the valves in the rooms
- The operator sets the desired temperature
- The operator turns the system on and off
- The controller gets notified of the new desired temperature
Attributes

<table>
<thead>
<tr>
<th>Thermostat</th>
</tr>
</thead>
<tbody>
<tr>
<td>desired-temp</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>On-Off switch</th>
</tr>
</thead>
<tbody>
<tr>
<td>setting</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Temp Sensor</th>
</tr>
</thead>
<tbody>
<tr>
<td>temperature</td>
</tr>
</tbody>
</table>
Iterate the Model

- Keep on doing this until you, your customer, and your engineers are happy with the model