


Conceptual Modeling

A Short Discussion


06-Modeling 1



Outline

- Conceptual modeling
 - The goal of conceptual modeling
- The OO solution
- The object model (conceptual)
 - Syntax and semantics
- Object modeling approach
 - Home Heating System


06-Modeling 2



Conceptual Modeling

- Early modeling to understand the problem
- Conducted in cooperation with the customer
 - Domain experts
 - ◆ Domain engineers
- No real problem analysis if the customer is not involved
- Power of OO
 - It is simple and people can quickly participate effectively


06-Modeling 3




The OO Solution

- The OO model closely resembles the problem domain
 - Base your model on the objects in the problem domain
- Iteratively refine the high-level model until you have an implementation
 - Attempt to avoid big conceptual jumps during the development process

06-Modeling 4




Objects



06-Modeling 5

Attributes and Operations

Person objects



abstracts to

Person class

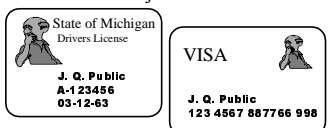
Attributes

- name
- age
- height
- weight

Operations

- move
- change-job

Card objects



Card class

Attributes

- height
- width
- id-number

Operations

- issue
- change

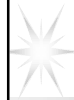


Characteristics of Objects

- Identity
 - Discrete and distinguishable entities
- Classification
 - Abstract entities with the same structure (attributes) and behavior (operations) into classes
- Polymorphism
 - The same operation may behave differently on different classes
- Inheritance
 - Sharing of attributes and operations based on a hierarchical relationship

06-Modeling

7



The Class Diagrams

06-Modeling

8



Objects

- Something that makes sense in the application context (application domain)
 - J.Q. Public
 - Joe's Homework Assignment 1
 - J. Q. Public's drivers license
- All objects have identity and are distinguishable
- NOT objects
 - Person
 - Drivers license

06-Modeling

9



Classes

- Describes a group of objects with similar properties (attributes), common behavior (operations), common relationships to other classes, and common semantics
- Person
 - J. Q. Public
 - Joe Smith
 - D. Q. Public
- Card
 - Credit card
 - Drivers license
 - Teller card

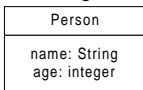
06-Modeling

10



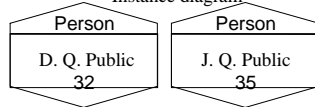
Class Diagrams

Class diagram

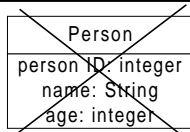


Class with attributes

Instance diagram



Objects with values



Objects have an identity

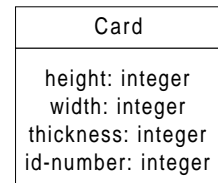
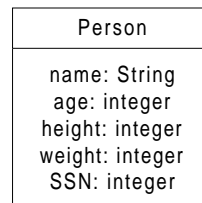
Do not explicitly list object identifiers
SSN OK!

06-Modeling

11



Examples



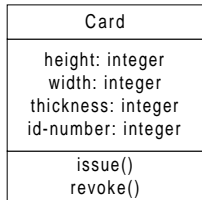
06-Modeling

12



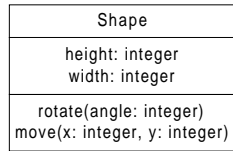
Operations and Methods

- Transformation that can be applied to or performed by an object
- May have arguments

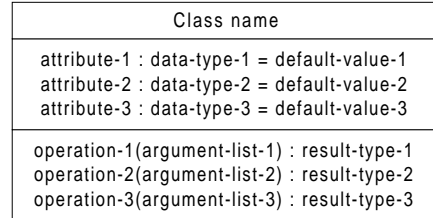


06-Modeling

13



Object Notation - Summary



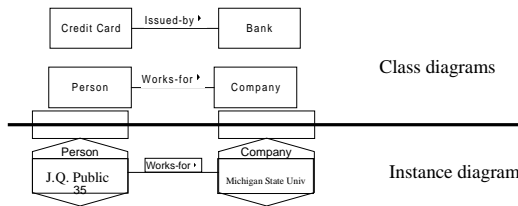
06-Modeling

14



Associations

- Conceptual connection between classes
 - A credit card is issued-by a bank
 - A person works-for a company



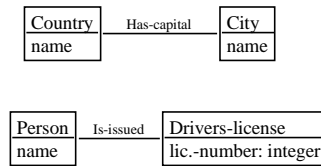
06-Modeling

15



Associations are Bi-directional

- There is no direction implied in an association



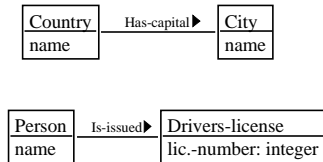
06-Modeling

16



Associations Have Direction

- Unified adds a direction indicator
 - Inconsistently used



06-Modeling

17



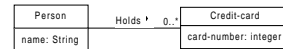
Multiplicity

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 (* stands for many)



06-Modeling

18

Multiplicity (Cont.)

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

06-Modeling 19

Multiplicity (Cont.)

- * 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
- Driver's License is optional (0..1)

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

06-Modeling

Higher order associations

- Ternary association
 - Project, language, person
- Seldom needed (and should be avoided)

06-Modeling 21

Link Attributes

- Associations can have properties the same way objects have properties

How to represent salary and job title?

Use a link attribute!

06-Modeling 22

Folding Link Attributes

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

06-Modeling 23

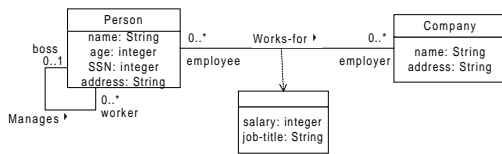
Another Approach

06-Modeling 24



Role Names

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



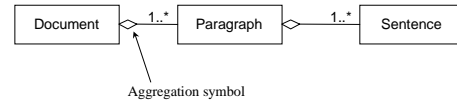
06-Modeling

25



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)



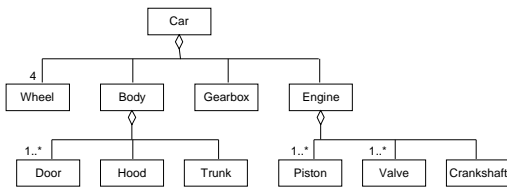
06-Modeling

26



Aggregation (Cont.)

- Often used in parts explosion



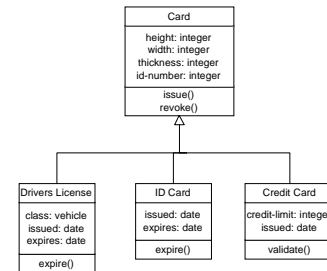
06-Modeling

27



Generalization and Inheritance

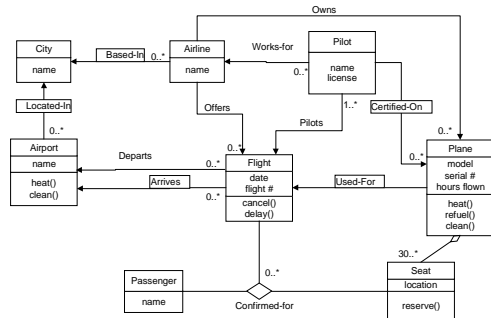
- The is-a association
 - Cards have many properties in common
 - Generalize the common properties to a separate class, the base-card
 - Let all cards inherit from this class, all cards is-a base-card (plus possibly something more)



06-Modeling

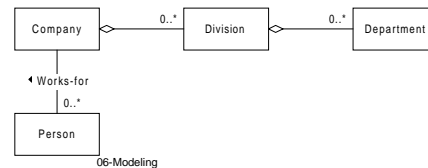
28

Example



Aggregation Versus 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.....



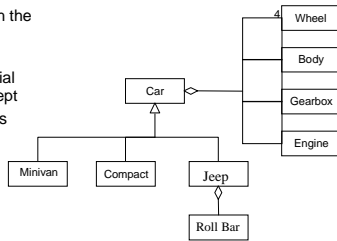
06-Modeling

30



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



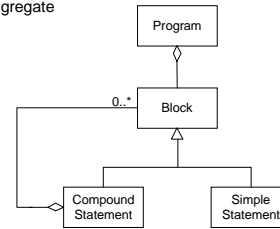
06-Modeling

31



Recursive Aggregates

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



06-Modeling

32



Object Modeling Summary

- Classes
 - Name
 - Attributes
 - Operations
- Associations
 - Roles
 - Link attributes
- Aggregation
- Inheritance

06-Modeling

33



Object Modeling Approach

06-Modeling

34



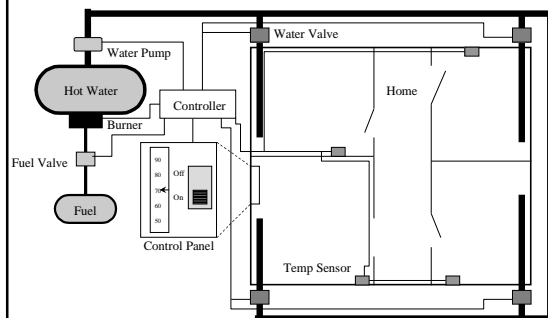
Object Modeling Approach

- 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

06-Modeling

35

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
- 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

06-Modeling 37

Identify Object Classes

Requirements Statements → Extract Nouns → Tentative Object Classes → Eliminate Spurious Classes → Object Classes

Candidate Classes

06-Modeling 38

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
- 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

06-Modeling 39

Eliminate Classes

Redundant: heating system, user

Irrelevant: Fuel, software, Hot Water

Vague: heat, house, heat flow, home, range

Attributes: desired temp, temperature

Operations: None

Roles: None

Implementation: None

06-Modeling 40

Classes After Elimination

06-Modeling 41

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

06-Modeling 42



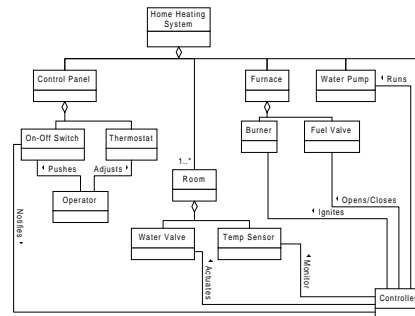
Possible Associations

- 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

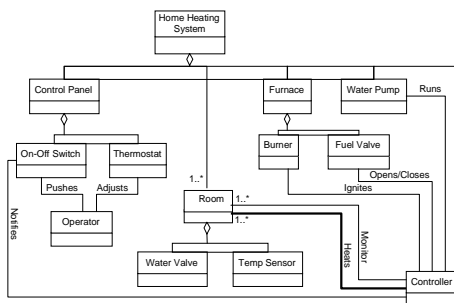
06-Modeling

43

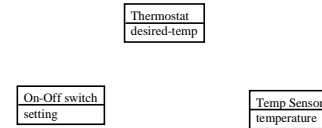
Object Model



Object Model - Modified



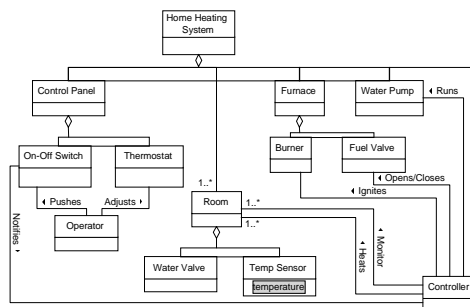
Attributes



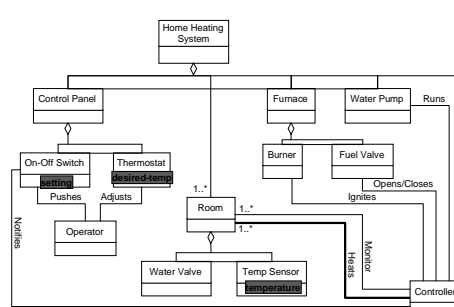
06-Modeling

46

Final OO Model



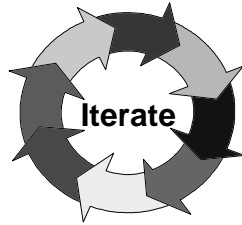
Final Object Model





Iterate the Model

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



06-Modeling

49



Operation vs Method

- **Operation:** specifies object behavior
- **Operations** of class are public **services** offered by class.
- **Service:** represented by set of operations.
- **Message:** object requests execution of an operation from another object by sending it message.
- **Method:** message is matched up with method defined by the class to which the receiving object belongs (or any of its superclasses)
- **Methods** of its classes are the implementations of these **operations**.

06-Modeling

50