System Modeling

Chapter 5

System Modeling

Outline:

I. What are system models?

II. System models:

1. Simple context diagram
2. UML class diagram
3. UML state diagram
4. Control flow diagram

I. System Modeling

- System modeling is
  - the process of developing abstract representations of a system
  - each model presents a different perspective of that system.
    - static: represents structure
    - dynamic: represents behavior
- System models are **Abstract**
  - Not an alternate representation
  - Some details are left out

System Modeling

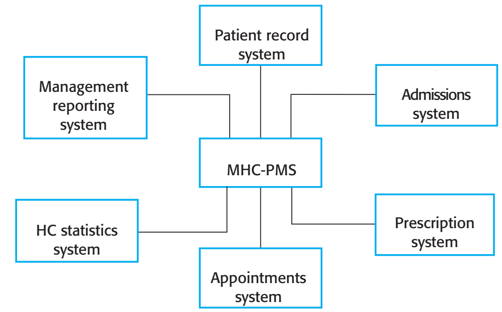
- Models of the system are used in:
  - Requirements development
    - clarification, discussion
  - Design process
    - represent plans for implementation
- Models discussed in this class:
  - Use case diagrams (ch. 4)
  - Architectural design diagrams (ch. 6)
  - Simple context diagrams (SRS)
  - UML class diagrams (SRS)
  - UML state diagrams
  - Control flow diagrams

UML=Unified Modeling Language

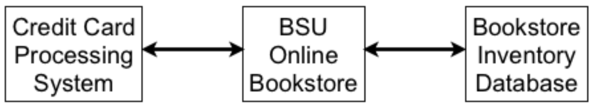
II - 1.Simple Context Model

- Used to define system boundaries
  - indicates what is done by the system being developed, and what will be done manually or by some other system
- Represented as a box and line diagram:
  - Boxes show each of the systems involved
  - Lines show interaction between systems
  - System being developed is in the center

Fig 5.1: The context of the MHC-PMS



From the BSU Online Bookstore SRS: Section 2.1 Product Perspective

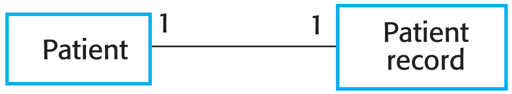


- Arrowheads not necessary
- Database is often NOT external
- Include a diagram like this in your SRS

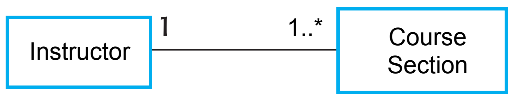
2. UML Class Diagrams

- Static model: represents structure, NOT behavior
- Shows object-oriented classes and associations between them
- Uses:
  - developing requirements: to model real-world objects
  - during design phase: add implementation objects
- Simple class diagrams:
  - **Box** represents a class (with a name)
  - **Lines** show associations between classes (name optional)
  - **Number** at each end to show how many objects can be involved in the association (multiplicity)

Fig 5.8: UML Classes and association

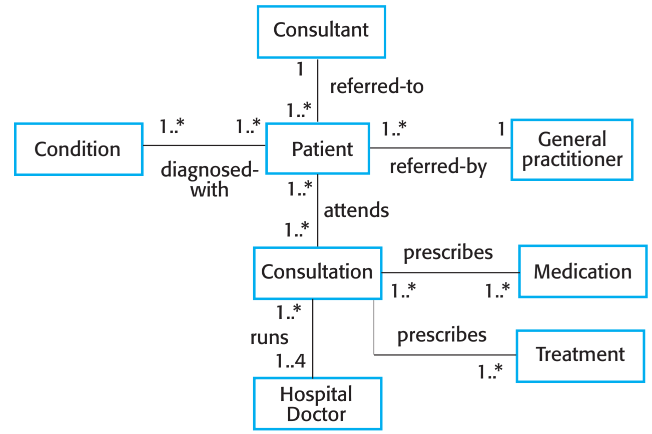


Two classes and one association (**a one-to-one relationship**)
- Each patient has 1 patient record
- Each patient record belongs to 1 patient



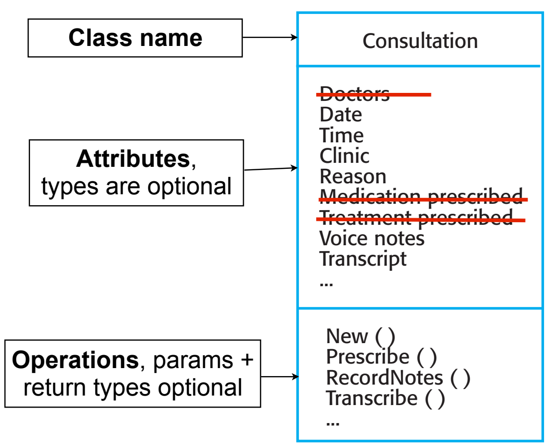
Two classes and one association (**a one-to-many relationship**)
- Each instructor teaches one or more course sections (1..*)
- Each course section is taught by exactly 1 instructor.

Fig 5.9: Classes and associations in the MHC-PMS



Condition - Patient is **a many-to-many relationship**
- Each patient has one or more condition
- Each condition may be had by one or more patients.

Fig 5.10: Consultation class, in more detail



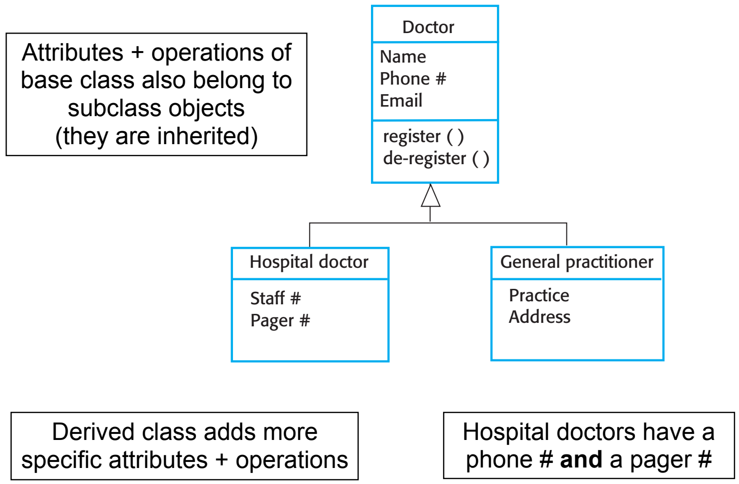
The class box may also specify the **attributes** and **operations** of the class.

Note: Don't record associated classes here (use arrows in the diagram)

Generalization (Inheritance)

- Act of identifying commonality among concepts, defining:
  - a general concept (base class)
  - specialized concept(s) (derived class).
- Common attributes are stored in superclass only
  - avoids duplication in diagram and code
- UML class diagram:
  - **Arrow** points from derived classes to base class
- Example: University personnel
  - Faculty, Staff, Students (graduate and undergrad)
  - All university personnel have ID numbers
  - All students have majors

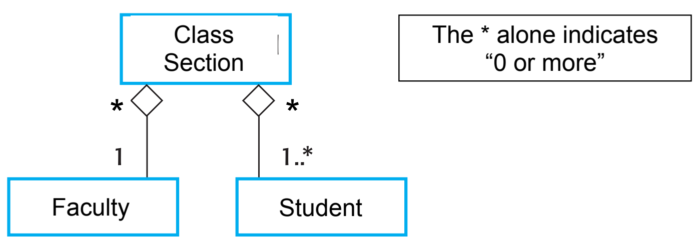
Fig 5.12: Generalization in UML class diagram



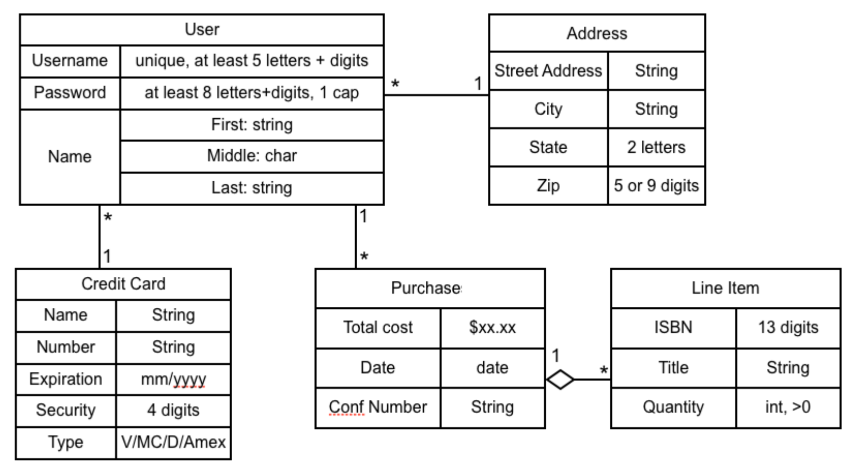
Aggregation (composition)

- When objects are composed of separate parts
  - ex: a university class is composed of a faculty member and several students
- UML class diagram:
  - diamond at end of line closest to “whole” class
- When should you use a diamond?
  - to represent that one object is a “part of” another
  - there is no formal definition.

Fig 5.13: Aggregation in UML class diagram



From the BSU Online Bookstore SRS: Section 3.4 Logical Structure of the Data



From the BSU Online Bookstore SRS: Section 3.4 Logical Structure of the Data

- Used to model “real world” objects during requirements engineering
- No operations indicated.
- Associations with multiplicity ARE indicated.
- Attribute types are NOT from C++, they are more specific and more descriptive.
  - Some include constraints
- Include a diagram like this in your SRS

3. UML State diagrams

- Dynamic model: represents behavior (not structure)
- Describes
  - all the <u>states</u> an (object or component or system) can get into
  - how state changes in response to specific events (<u>transitions</u>)
- Useful when object/component/system is changed by events (real time and embedded systems, etc.)
  - mouse click on certain element
  - certain button is pushed
  - sensor reports a certain value

UML State diagrams

- Components of a state diagram:
  - **Rounded rectangles:** system states
    - includes what action to **do** in that state
  - **Labeled arrow:** stimuli to force transition between states
    - **optional guard**: transition allowed only when guard is true
    - **unlabeled arrow:** transition occurs automatically when action is complete

Fig 5.16 State diagram of a microwave oven

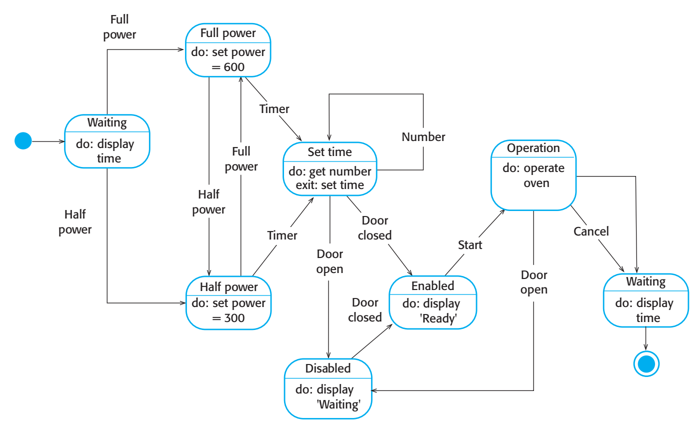


Diagram is missing (at least) one arrow

4. Control Flow diagrams
aka Flowcharts

- Dynamic model: represents behavior (not structure)
- Not a UML model (it's old school)
  - the UML Activity diagram can model same information
- Describes:
  - the flow of control through an algorithm or process
  - branching using diamonds to represent decision points
  - repetition or looping using "back arrows"

Control Flow diagrams

- Components of a control flow diagram:
  - **Rounded rectangles:** represent actions or processing
    - input/output, storing/retrieving values, computation
  - **Arrow:** shows flow of control, where to go next
    - may return to a previous action, forming a loop.
  - **Diamond**: contains yes/no question (or T/F)
    - has two arrows coming out of it, one labeled "yes", other labeled "no"
  - **Start and end:** rectangles indicating where algorithm starts and stops.

control flow diagram: example