System Modeling

Chapter 5
part B

System Modeling
in the textbook

- Context models
- Interaction models
- **Structural models**
- **Behavioral models**
- **Model-driven engineering**

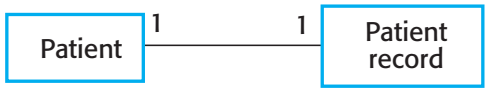
5.3 Structural Models

- Display the organization of a system in terms of its components and relationships
- Static Models
  - shows structure of system design
- Dynamic Models
  - shows organization of system when it is executing (processes/threads)
  - (won't be discussing these)

5.3.2 UML Class Diagrams

- Static model
- Shows classes and associations between them
- Uses:
  - developing requirements: model real-world objects
  - during design phase: add implementation objects
- Simple class diagrams:
  - **Box** represents a class (with a name)
  - **Lines** show associated between classes
  - **Number** at each end to show how many objects can be involved in the association

Fig 5.8: UML Classes and association

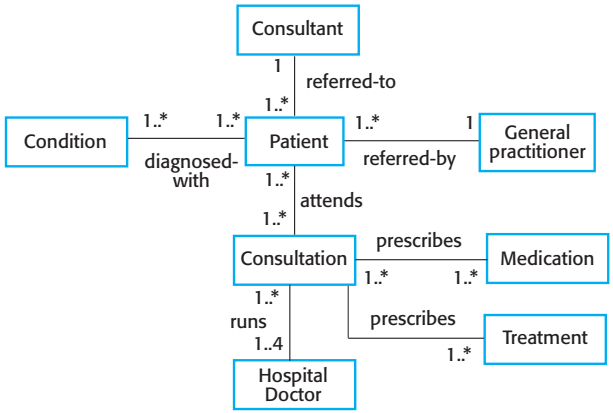


Two classes and one association
(a one-to-one relationship)

UML Class Diagrams

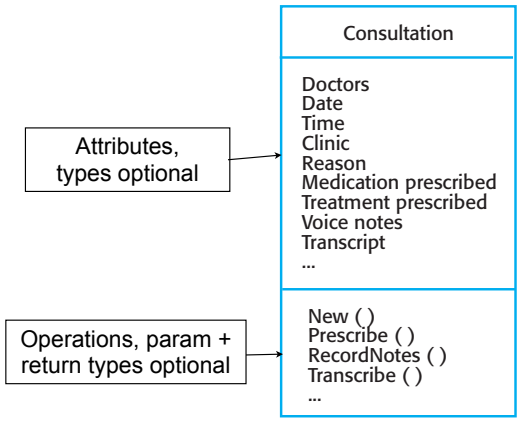
- An object is an instance of a class
- Associations can be named
- Now resembles a Semantic data model:
  - used in database design
  - ex: an ER diagram (entity-relationship)
- But you cannot have attributes of relationships
  - unless you make the relationship into a class

Fig 5.9: Classes and associations in the MHC-PMS



Note: 1..* indicates “one or more”

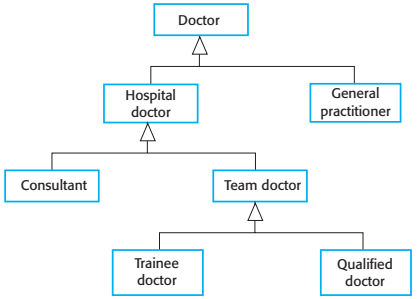
Fig 5.10: Consultation class, in more detail



5.3.2 Generalization

- Act of identifying commonality among concepts, defining:
  - a general concept (superclass)
  - specialized concept(s) (subclasses).
- Example: University personnel
  - Faculty, Staff, Students (graduate, undergrad)
  - All university personnel have ID numbers
  - All students have majors
- Common attributes are stored in superclass only
  - change affecting ID number happens in University personnel class only

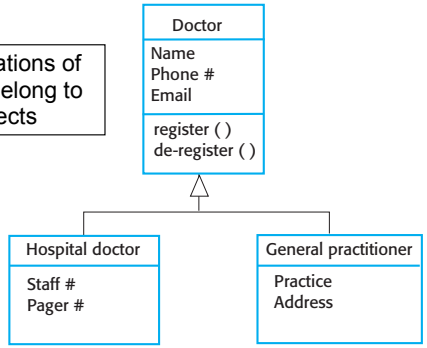
Fig 5.11: Generalization hierarchy



Arrow points to superclass

Fig 5.12: Generalization with added detail

Attributes + operations of superclass also belong to subclass objects

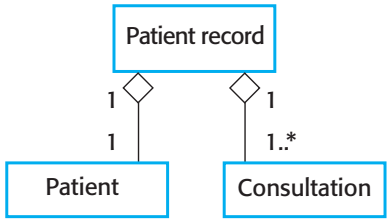


Subclass adds more specific attributes + operations

5.3.3 Aggregation

- When objects are composed of separate parts
  - ex: a (university) class is composed of a faculty member and several students
- UML: aggregation is a special kind of association
  - diamond at end of line closest to “whole” class
- When implemented, the composite usually has instance variables for each “part” object

Fig 5.13: Aggregation association



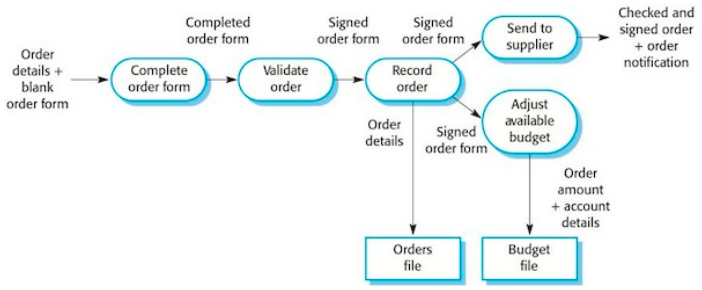
5.4 Behavioral models

- Represent dynamic behavior of the system as it is executing,
- More of an "internal" view of the system
- Sequences of Actions:
  - UML Activity diagrams (process, flow of actions)
  - UML Sequence diagrams (sequence of interactions)
  - **Data-flow diagrams (DFD)**
- States of an object or system, with transitions
  - **UML state diagrams**

5.4.1 Data-flow diagram

- illustrate how data is processed by a system in terms of inputs and outputs.
- Among the first graphical software models (not UML)
- Models sequence of actions in a process
  - sequence of functions, each with input and output data
  - functional or procedural -oriented (not objects)
- Useful during requirements analysis:
  - simple and intuitive, users can validate proposed system

Example Data Flow Diagram: Order Processing



Oval: functional processing
Rectangle: data store
Labeled arrow: data and movement

5.4.2 UML State diagrams

- Describes
  - all the states an (object or component or system) can get into
  - how state changes in response to events (transitions)
- Useful when object/component/system is changed by events (real time and embedded systems, etc.)
- Components of a state diagram
  - **Rounded rectangles:** system states
    - includes what action to **do** in that state
  - **Labelled 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

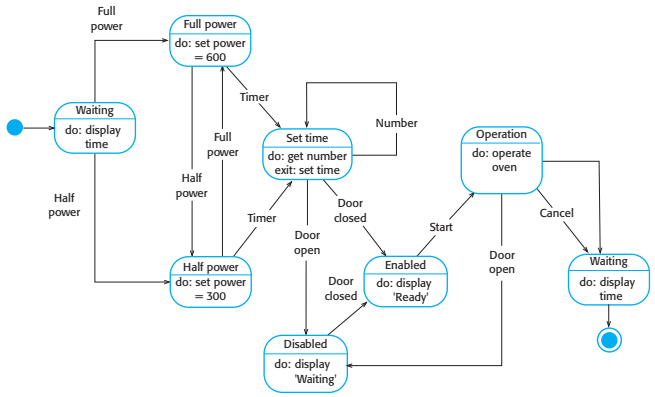


Diagram is missing (at least) one arrow

5.5 Model Driven Engineering (MDE)

- An approach to software development where models (rather than programs) are the principal outputs of the development process.
  - Developers generate programs automatically from the models.
  - Developers test and debug models rather than programs
- Models are often extensions of UML models
- Some problems:
  - Models are inherently too abstract to be a basis for the implementation.
  - Not enough good tools supporting model compilation and debugging yet.