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)
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.

Class name
Attributes, types are optional
Operations, params + return types optional

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

Attributes + operations of base class also belong to subclass objects (they are inherited)

- Doctor
  - Name
  - Phone #
  - Email
  - register ()
  - de-register ()

- Hospital doctor
- General practitioner
  - Practice
  - Address

- Staff #
- Pager #

Derived class adds more specific attributes + operations

Hospital doctors have a phone # and a pager #

Fig 5.13: Aggregation in UML class diagram

The * alone indicates “0 or more”

Class Section

- Faculty
- Student

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.

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

<table>
<thead>
<tr>
<th>User</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
<td>Street Address</td>
</tr>
<tr>
<td>Password</td>
<td>City</td>
</tr>
<tr>
<td></td>
<td>State</td>
</tr>
<tr>
<td>Name</td>
<td>Zip</td>
</tr>
<tr>
<td>First: string</td>
<td>5 or 9 digits</td>
</tr>
<tr>
<td>Middle: char</td>
<td></td>
</tr>
<tr>
<td>Last: string</td>
<td></td>
</tr>
<tr>
<td>Credit Card</td>
<td>Purchase</td>
</tr>
<tr>
<td>Name</td>
<td>Total cost</td>
</tr>
<tr>
<td>Number</td>
<td>$xx.xx</td>
</tr>
<tr>
<td>Expiration</td>
<td>Date</td>
</tr>
<tr>
<td>Security</td>
<td>Conf Number</td>
</tr>
<tr>
<td>Type</td>
<td>String</td>
</tr>
<tr>
<td>V/MC/Discover</td>
<td>Title</td>
</tr>
<tr>
<td>Master Card</td>
<td>Quantity</td>
</tr>
<tr>
<td>Card</td>
<td>int, &gt;0</td>
</tr>
</tbody>
</table>

* The * alone indicates “0 or more”
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 states an (object or component or system) can get into
  - how state changes in response to specific events (transitions)
- 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