System Modeling

Chapter 5

System Modeling

- System modeling is
  - the process of developing abstract representations of a system
  - each model presents a different perspective of that system.

- System models are **Abstract**
  - Not an alternate representation
  - Some details are left out
  - Incomplete

System Perspectives

Different perspectives presented by models:

- **external (or context)**: shows context or environment of the system

- **interaction**: shows interaction
  - between the system and its environment, or
  - between components within the system

- **structural**: shows organization of the system or structure of data

- **behavioral**: shows dynamic behavior, including how the system responds to events

System Modeling in the textbook

- 5.1 Context models
- 5.2 Interaction models
- 5.3 Structural models
- 5.4 Behavioral models
- 5.5 Model-driven engineering
# System Modeling

- Notation used to represent the models:
  - Graphical (diagrams)
    - UML=Unified Modeling Language
  - Formal/mathematical (ch 12)
- Models of the system are used in:
  - Requirements development
    - clarification, discussion
  - Design process
    - represent plans for implementation
  - Model-driven engineering
- Precision and completeness: not always necessary

---

# UML Diagrams

We'll discuss these UML Diagrams

- **Activity diagrams**: the activities in a process.
- **Use case diagrams**: interactions between a system and its environment.
- **Sequence diagrams**: interactions between actors and the system and components.
- **Class diagrams**: classes in the system and the associations between these classes.
- **State diagrams**: how the system reacts to events.

---

## 5.1 Context Models

- Primarily an external perspective
  - shows how the system is situated or involved in its context
- Two sub-views within the perspective:
  - **Static view**: shows what other systems the system will interact with
  - **Dynamic View**: shows how the system is involved in business processes

---

## Simple Context Model

**Static view**

- Used to define system boundaries
  - determines what is done by the system, and what will be done manually or by some other system
  - stakeholders must decide on this early
- Represented as a box and line diagram:
  - Boxes show each of the systems involved
  - Lines show interaction between systems
  - Technically NOT a UML diagram
Fig 5.1: The context of the MHC-PMS

Note: <<system>> is an example of a “stereotype” in UML
A mechanism to categorize an element in some way

UML Activity Diagram
Dynamic view

- Used to show how the system is used in business processes
- Shows activity and flow of control

- filled circle: start
- filled concentric circle: finish
- rounded rectangles: activities
- rectangles: other objects (i.e., the different systems in Fig 5.2)
- arrows: flow of work
- diamonds: branch (and merge)
- guards: condition under which flow is taken out of branch
- solid bar: activity coordination/concurrency control (fork, join)

Fig 5.2: Process model of involuntary detention

Example of a UML Activity diagram

Note: This diagram is missing one branch and 2 merge diamonds

5.2 Interaction Models

- These model interactions
  - between the system and environment or users
  - between components within the system

- Uses:
  - between user and system: developing requirements
  - between system components: help to understand flow of control in an object-oriented system, used in design

- UML Use Case Diagrams:
  - represent user-system interactions

- UML Sequence Diagrams:
  - represent interactions between components (and actors)
5.2.1 Use Case Modeling

- **Main purpose:** requirements elicitation + analysis

- **Use Case:** overview of one user/system interaction
  - Focused on one goal of the actor

- **Use Case Diagram components:**
  - **stick figure:** actor (user or system)
  - **ellipse:** named interaction (verb-noun)
  - **line:** indicates involvement in interaction

- **Diagram is supplemented with further details describing the use case:**
  - simple textual description or
  - structured description (form/template/table) or
  - sequence diagram(s)

Note: arrows are not part of UML, but shows direction of data flow

**Fig 5.3: Transfer data use case**

**Example of a UML Use case diagram**

**Fig 5.4: Tabular description of Transfer data use case**

<table>
<thead>
<tr>
<th>MHC-PMS: Transfer data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actors</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Data</strong></td>
</tr>
<tr>
<td><strong>Stimulus</strong></td>
</tr>
<tr>
<td><strong>Response</strong></td>
</tr>
<tr>
<td><strong>Comments</strong></td>
</tr>
</tbody>
</table>

**Fig 5.5: Use cases involving Medical Receptionist**

A composite use case diagram: all interactions involving a given actor
5.2.2 Sequence Diagram

- Models the interactions between actors and/or objects or components within the system in detail.
- Can be used to show the sequence of interactions in a given use case.
- Diagram notes:
  - Read sequence from top to bottom: it's chronological.
  - Objects and actors: listed across top with dotted lines going down.
  - Boxes on dotted line: lifetime of operation (in this interaction).
  - Dotted arrows between lines from objects: interactions.
  - Annotations on arrows: calls to objects with parameters, return values.
  - Box named alt with conditions in brackets: for branching/alternatives.

Fig 5.6: View patient information

Example of a UML Sequence diagram

Sequence Diagram Uses

- Requirements Development:
  - To document/discuss requirements (especially operations).
  - These diagrams must leave out detail (objects).
    - so as not to constrain developers.
  - For example:
    - Minimal sequence diagram: only two components: user and system.
    - Use to show sequence of interactions between user and system.

- Design/Implementation:
  - Details are required:
    - Messages must match objects’ methods.
    - Include arguments in method calls between objects.
    - Source of the arguments should be indicated.

5.3 Structural Models

- Display the organization of the system in terms of its components and relationships.

  - Static Models
    - shows the structure of the system.
    - or just the structure of the data.

  - 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 association 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

- **Patient**
  - 1:1
  - **Patient record**
  - Two classes and one association (a one-to-one relationship)

- **Instructor**
  - 1:1..*
  - **Course Section**
  - Two classes and one association (a one-to-many relationship)
  - How many instructors does a Course Section have?

Fig 5.9: Classes and associations in the MHC-PMS

- **Consultant**
  - referred-to
  - referred-by
  - attends
  - 1..*

- **Patient**
  - diagnosed-with
  - 1..*

- **General practitioner**
  - 1..*

- **Condition**
  - 1..*

- **Consultation**
  - 1..*

- **Medication**
  - prescribes
  - 1..*

- **Treatment**
  - prescribes
  - 1..*

- **Hospital**
  - runs
  - 1..4

- **Doctor**
  - 1..*

Fig 5.10: Consultation class, in more detail

- **Consultation**
  - Attributes, types optional
  - Operations, param + return types optional

- **Doctors**
  - Note: Don’t record associated classes here
  - Date
  - Time
  - Clinic
  - Reason
  - Medication prescribed
  - Treatment prescribed
  - Voice notes
  - Transcript
  - ...

- **New ()**
  - Prescribe ()
  - RecordNotes ()
  - Transcribe ()
  - ...

- Note: Don’t record associated classes here
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
  - avoids duplication
  - changes affecting how ID number is implemented 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 (they are inherited)
  - Name
  - Phone #
  - Email
  - register ()
  - de-register ()

- 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
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 (flow of actions, slide 10)
  - UML Sequence diagrams (sequence of interactions, slide 17)
  - Data-flow diagrams (DFD)
- States of an object or system, with transitions
  - UML state diagrams

5.4.1 Data-flow diagram

- Many systems are (were?) data-processing systems, primarily driven by data.
  - One of the first graphical software models (pre UML)
- DFD models the flow of data through a process
  - Collection of connected functions, each with input and output data
  - Shows dependencies of functions
  - More functional or procedural-oriented
- Useful during requirements analysis:
  - Simple and intuitive, users can validate proposed system
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
  - **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

![State diagram of a microwave oven](image)

Fig 5.16
State diagram of a microwave oven

Diagram is missing (at least) one arrow

5.5 Model Driven Engineering (MDE)

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