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

System Perspectives

Different perspectives presented by models:

• **external**: context or environment of the system

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

• **structural**: organization of the system, or structure of data

• **behavioral**: dynamic behavior, including how the system responds to events
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
Note: <<system>> is an example of a “stereotype” in UML
A mechanism to categorize an element in some way

Process Model
Dynamic view

- Shows how the system is used in business processes
- Represented as a UML Activity diagram
  - Shows activity and flow of control

filled circle: start
filled concentric circle: finish
rounded rectangles: activities
rectangles: other objects (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)

5.2 Interaction Models

- Model interactions
  - between the system and environment or users
  - between components within the system
- Uses:
  - user and system: developing requirements
  - system components: help to understand flow of control in an object oriented system
- Use Case Diagrams:
  - represent user-system interactions
- Sequence Diagrams:
  - represent interactions between components (and actors)
5.2.1 Use Case Modeling

- Main purpose: requirements elicitation + analysis
- Overview of one discrete 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
  - simple textual description or
  - structured description (form/template/table) or
  - sequence diagram(s)

Fig 5.3: Transfer data use case

Example of a UML Use case diagram

![Transfer data use case diagram](image)

Medical receptionist → Transfer data → Patient record system

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

Note: primary actor on left, supporting actor on right

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

![Use cases involving Medical Receptionist](image)

Medical receptionist → Register patient → Unregister patient → View patient info. → Transfer data → Contact patient
5.2.2 Sequence Diagram

- Models the interactions between actors and objects within the system in some detail
- Can be used to show the sequence of interactions in a given use case
- Diagram notes:
  Read sequence from top to bottom
  objects and actors: listed across top with dotted lines going down
  boxes on dotted line: lifetime of object (in this interaction)
  dotted arrows between lines from objects: interactions, messages
  annotations on arrows: calls to objects with parameters, return values
  box named alt with conditions in brackets: for branching/alternatives

Sequence Diagram Uses

- Requirements Development:
  - To document/discuss requirements
  - These diagrams must leave out detail
    - 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 parameters in method calls between objects
  - Source of the parameters

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
  - 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 (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

![Diagram showing two classes and one association](image)

How many instructors does a Course Section have?

Fig 5.9: Classes and associations in the MHC-PMS

![Diagram showing classes and associations](image)

Fig 5.10: Consultation class, in more detail

![Diagram showing consultation class and its attributes and operations](image)

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

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 (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

- Many systems are data-processing systems, primarily driven by data.
- DFD illustrates how data is processed by the system in terms of inputs and outputs.
  - One of 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 (input/output) 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.