CS 342 Software Design

- System modeling
- UML
- Make your own UML

Modeling

Modeling is the designing of software projects before coding.

System modeling

What is system modeling?

The process of developing abstract models of a system where each model represents a different view or perspective of that system.

Why do we use system modeling?

System modeling can be used as a planning tool, a road map for what you are going to do.

Models help us understand the functionality, scope, and requirements of a system and are used to communicate these things to others involved in the process (customers and other team members).

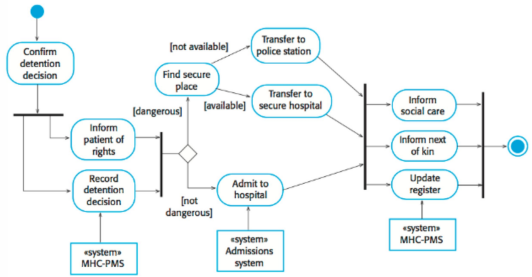
Models also serve as important documentation of the project

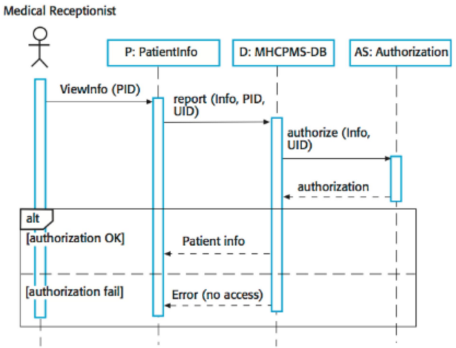
Different models help explain the system from different perspectives.

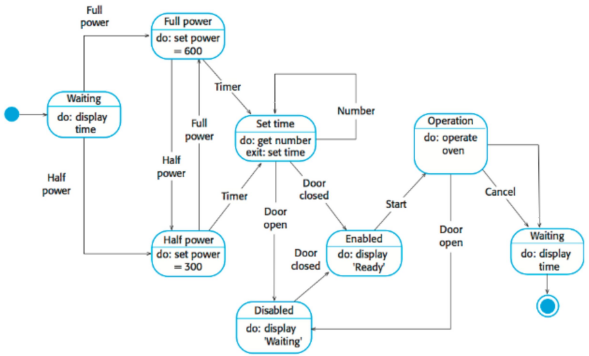
- **Externals:** The context or environment of the system.
- **Interaction:** Interactions between the system and its environment or interactions between different components of the system.
- **Structural:** The organization of a system or the structure of data processed by a system.
- **Behavioral:** The dynamic behavior of a system and how it responds to different events.

Examples of system modeling.

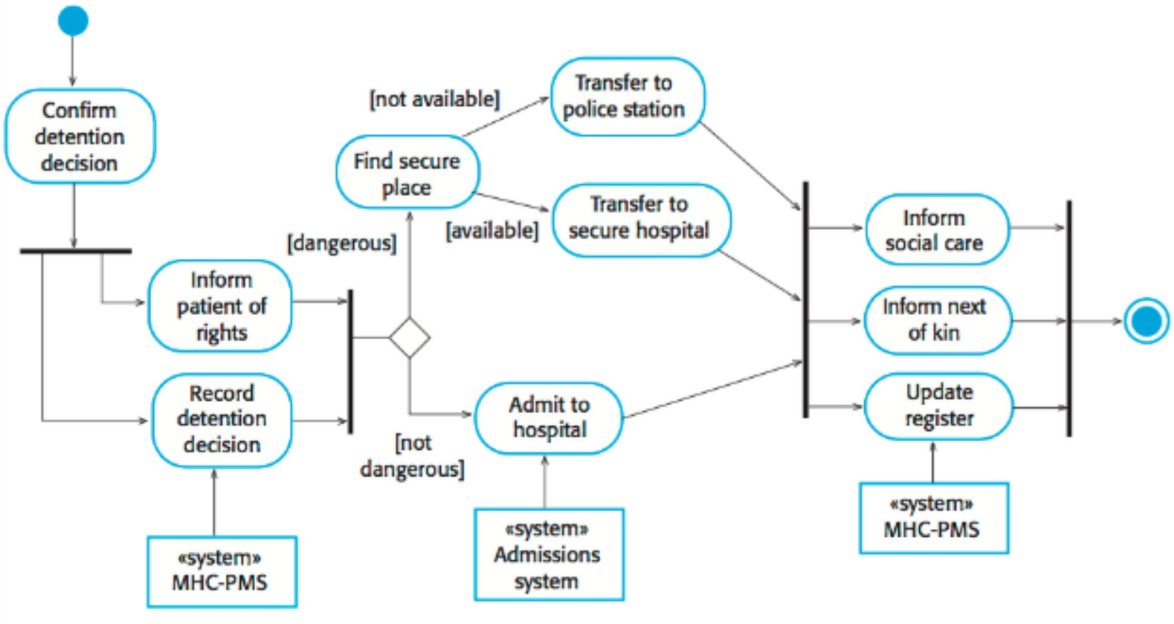
System modeling can be done in different ways. It can focus on elements of your solution that are important to engineers or elements of your solution that are important to other disciplines.





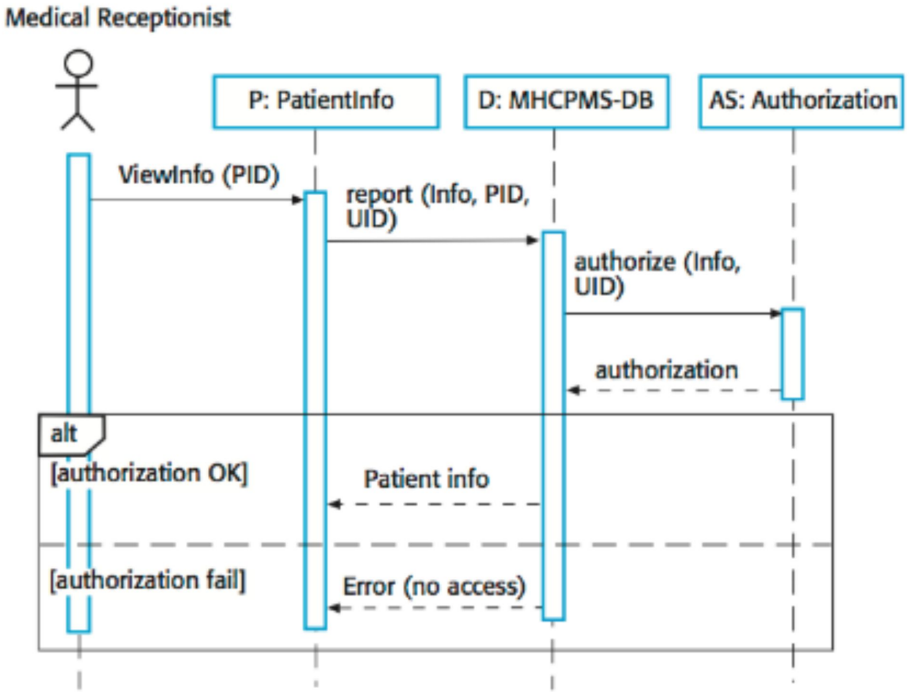


Activity diagram.



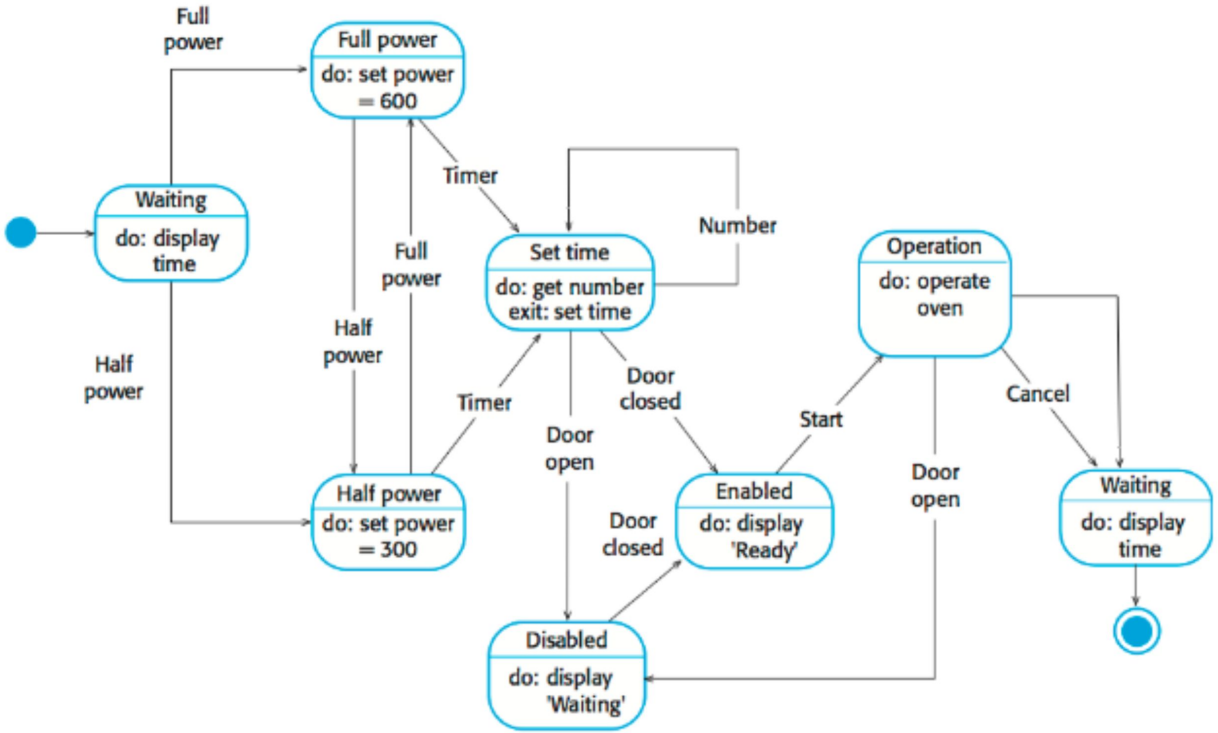
This **activity diagram** describes the process of involuntary detention and the role of MHC-PMS (mental healthcare patient management system) in it. The bubbles are events in the system, the bold lines indicate things that happen at the same time, the diamond is a choice (either this or that), the lines with arrows show the flow of events. The squares at the bottom show which part of the system is in charge at different points in the process.

Sequence diagram.



This **sequence diagram** shows actors and objects (parts of the system) at the top. The long bars indicate when in the process each actor and object is involved and the arrows show the order of events and interactions. This kind of model is used for interactions between actors and objects; usually modeling a specific use case (in this case, transferring of data).

State diagram.



This **state diagram** is used to model the dynamic behavior of a system as it reacts to different events. The bubbles are states that the system can be in. The lines with arrows indicate what can happen from that state and what states it can transition to. At either end you will see the circles, the filled in circle on the LEFT indicates the start of the system and the circle at the bottom right indicates a finished state. Keep in mind, you can have multiple end states.

Models can be created in UML.

All of these models are created using UML, or **Unified Modeling Language.**

UML is a set of symbols that are used to create different types of diagrams that help model systems and behaviors.

What are the benefits of using UML? If you define the symbols used in your diagram, anyone can read and understand them.

CS 342 Software Design

UML: Unified Modeling Language

Modeling is the designing of software projects before coding.

Modeling is the designing of software projects before coding.

What can we model?

- Class, object, component…..diagrams
- Behavior (including use case diagrams), activity and state diagrams
- Sequence, communication, timing…..diagrams

******UML 2.0 defines 13 types of diagrams divided up into 3 categories ******

UML: Class diagram

Three sections: Name, Attributes and Methods

- Name: Capitalized, add <<interface>> above if interface
- Attributes: Access modifier Name: Type
- Methods: Access modifier Name(parameters): return type

Access modifiers: + (public), - (private), # (protected), ~ (package)

Anything abstract is in italics

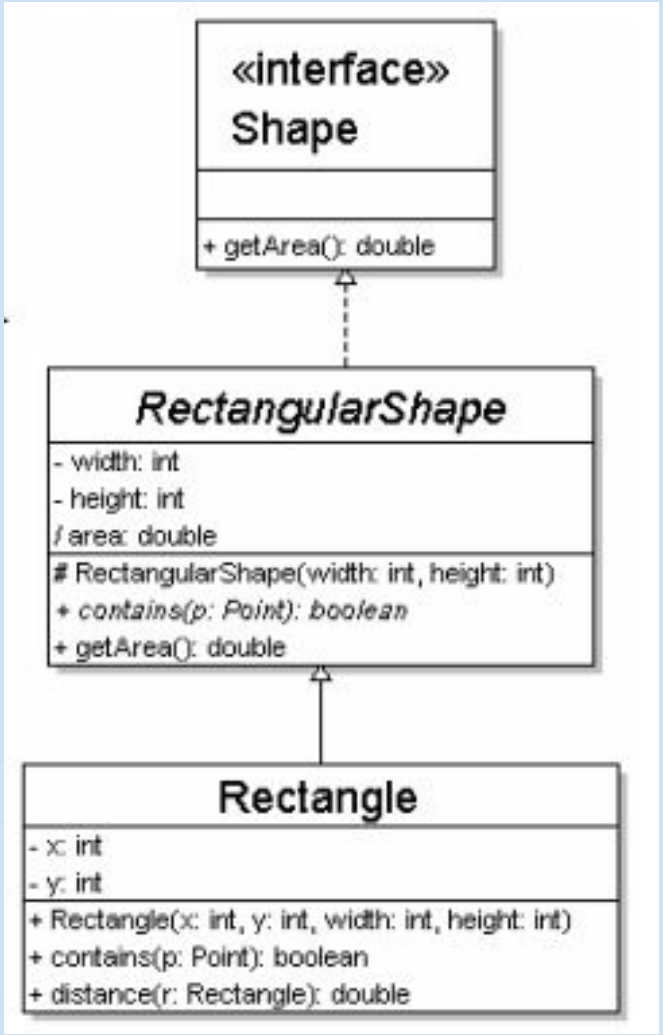
getters and setters usually omitted

underline static methods and attributes

UML: inheritance

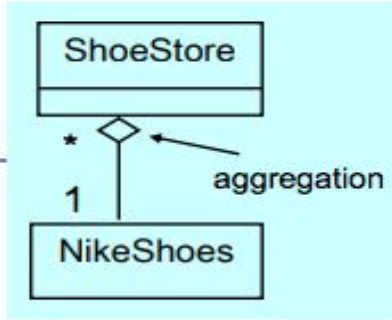
Use open triangle tipped arrows for inheritance

Use open triangle with dotted line arrows for interfaces

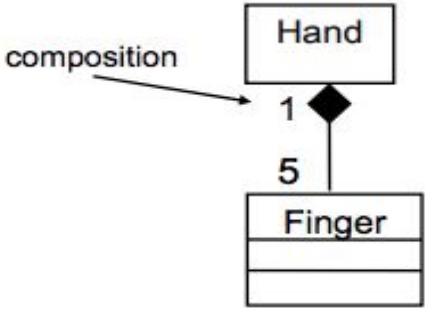


Association types

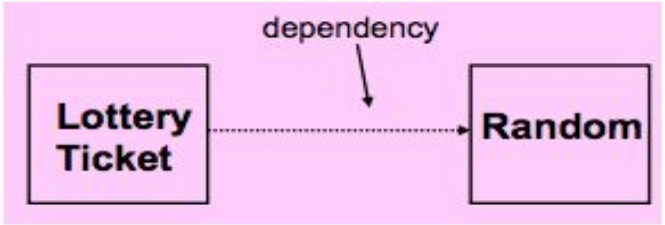
- **aggregation**: "contains"
  - symbolized by a clear white diamond pointing to *the class containing the other class*



- **composition**: "contained for only this purpose"
  - *stronger version of aggregation*
  - the parts live and die with the whole
  - symbolized by a black diamond pointing to the containing class



- **dependency**: "uses temporarily"
  - symbolized by dotted line



Now you try it!!!!

Create a class diagram of Project #1

https://app.creately.com/manage/project/home

http://www.umlet.com/umletino/umletino.html

UML: Project #1

- What classes would we need?
  - What methods and attributes would they need?
- What interfaces?
- How would they interact?
  - Inheritance?
- What type of associations would we have?