System Modeling

Software system modeling

- System models – Abstract descriptions of systems whose requirements are being analysed
- Objectives
  - Context models
  - Interaction models
  - Structural models
  - Behavioral models

Software modeling and models

- Software modeling helps the engineer to understand the functionality of the system
- Models are used for communication among stakeholders
- Different models present the system from different perspectives
  - An external perspective, where you model the context or environment of the system.
  - An interaction perspective, where you model the interactions between a system and its environment, or between the components of a system.
  - A structural perspective, where you model the organization of a system or the structure of the data that is processed by the system.
  - A behavioral perspective, where you model the dynamic behavior of the system and how it responds to events.

The Unified Modeling Language (UML)

Devised by the developers of OO analysis and design methods

- Context Models(context diagrams)
- Interaction Models(Use case diagrams)
- Structure Models(Class diagrams)
- Behavior Models(activity diagrams, sequence diagrams)

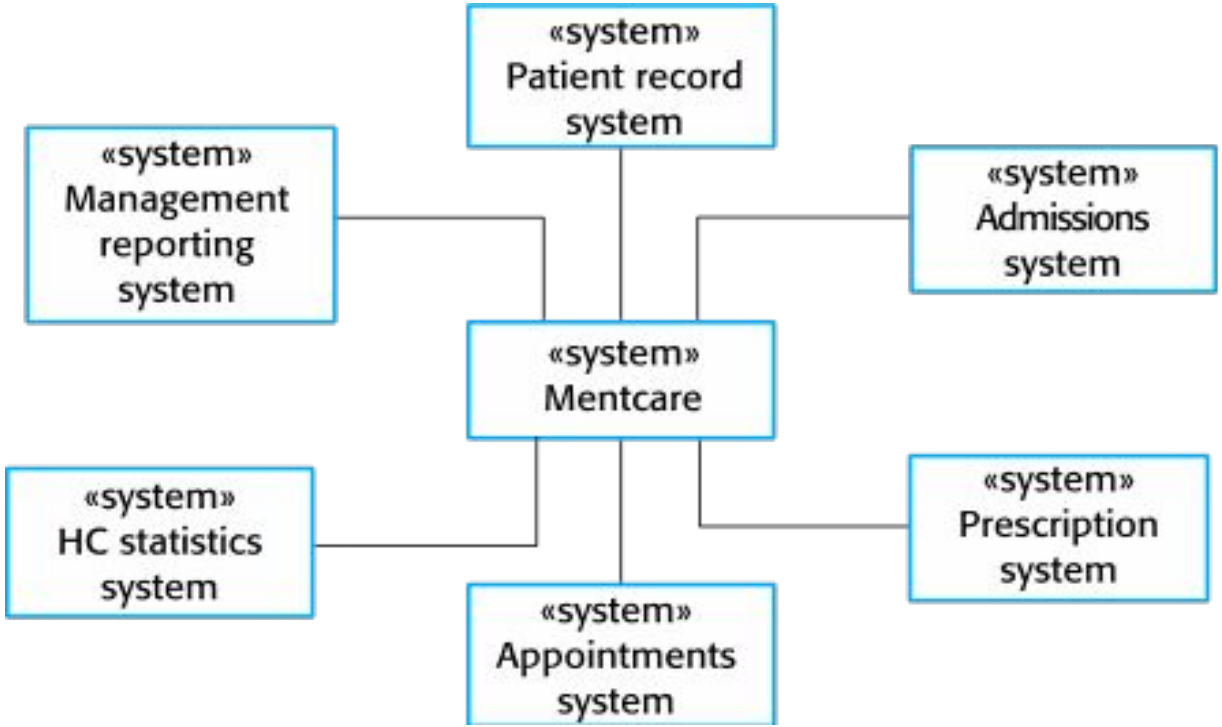
Use of Graphical Models

- As a means of facilitating discussion about an existing or proposed system
  - Incomplete and incorrect models are OK as their role is to support discussion.
- As a way of documenting an existing system
  - Models should be an accurate representation of the system but need not be complete.
- As a detailed system description that can be used to generate a system implementation
  - Models have to be both correct and complete.

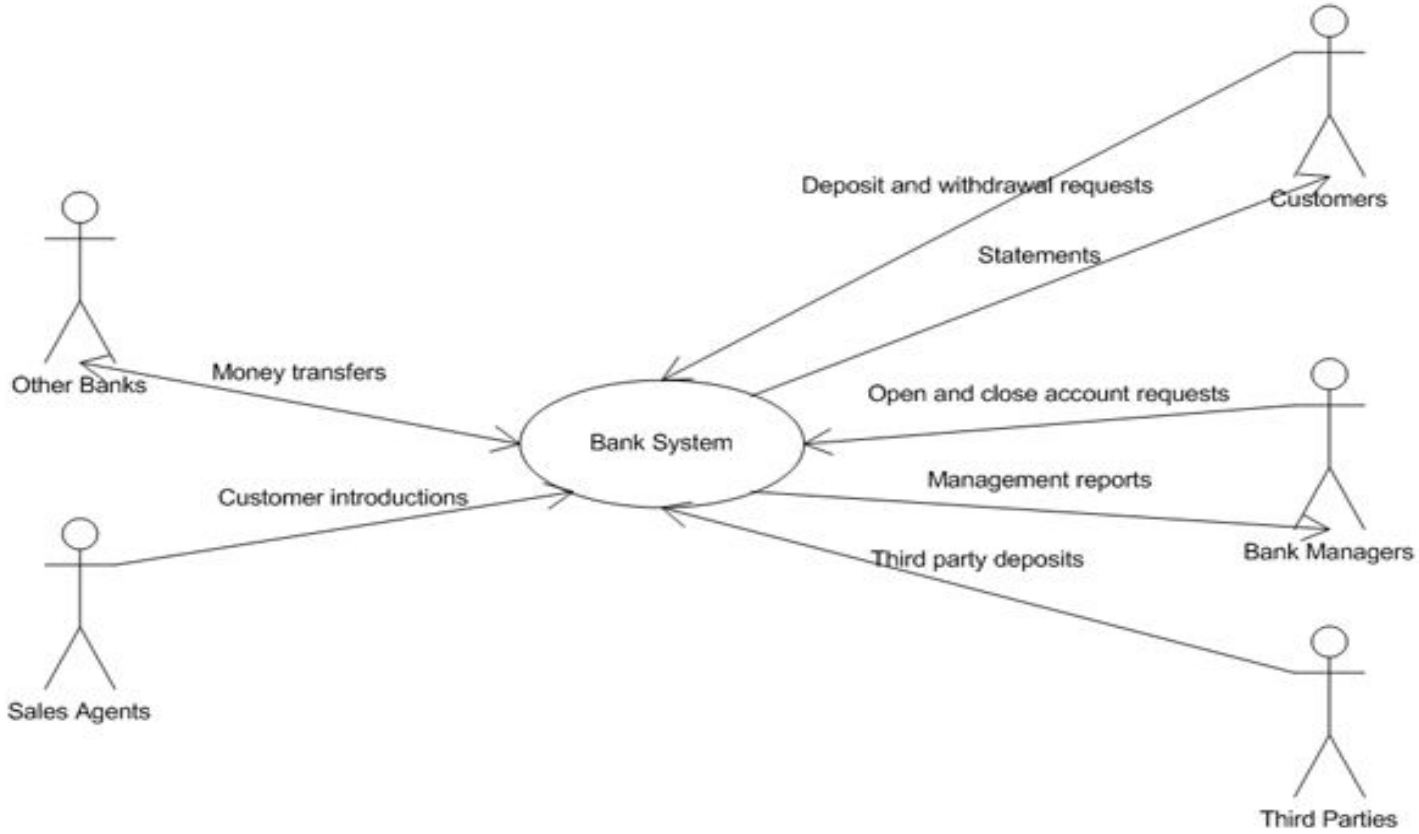
Context Models

- Context models are used to illustrate the operational context of a system - they show what lies outside the system boudaries.
- Social and organizational concerns may affect the decision on where to position system boundaries.
- Architectural models show the system and its relationship with other systems.

Context of Mentcare system



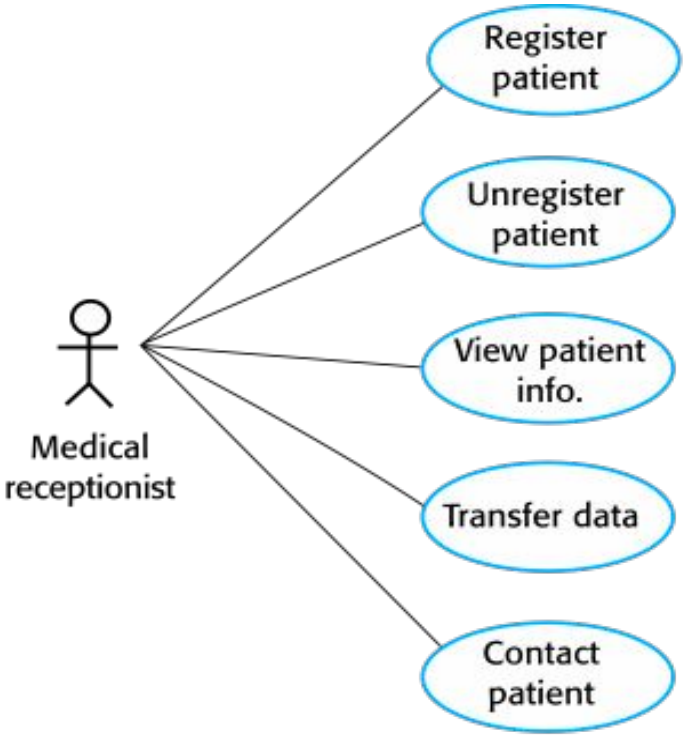
Context model



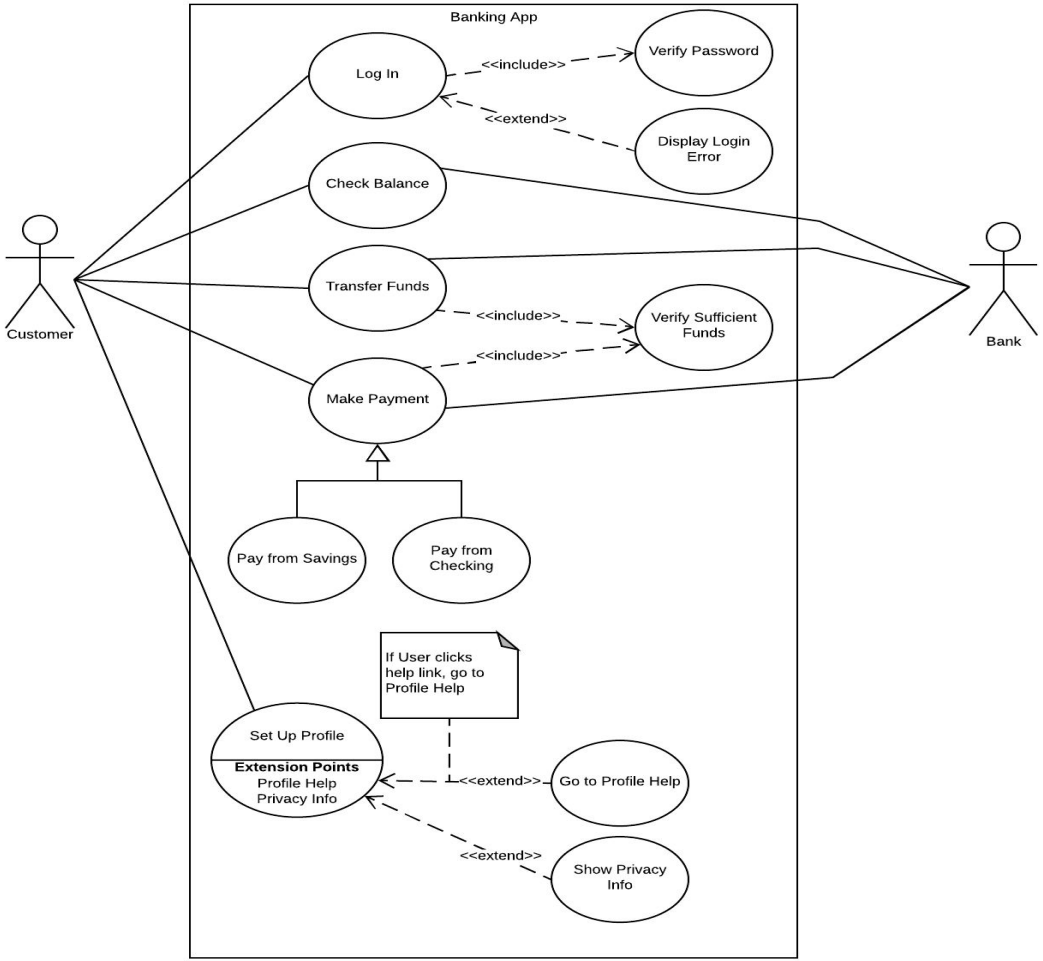
Interaction models

- Modeling user interaction is important as it helps to identify user requirements.
- Modeling system-to-system interaction highlights the communication problems that may arise.
- Modeling component interaction helps us understand if a proposed system structure is likely to deliver the required system performance and dependability.
- Use case diagrams and sequence diagrams may be used for interaction modeling.

Use cases in the Mentcare System involving the role ‘Medical receptionist’



Use case diagram



Structural Models

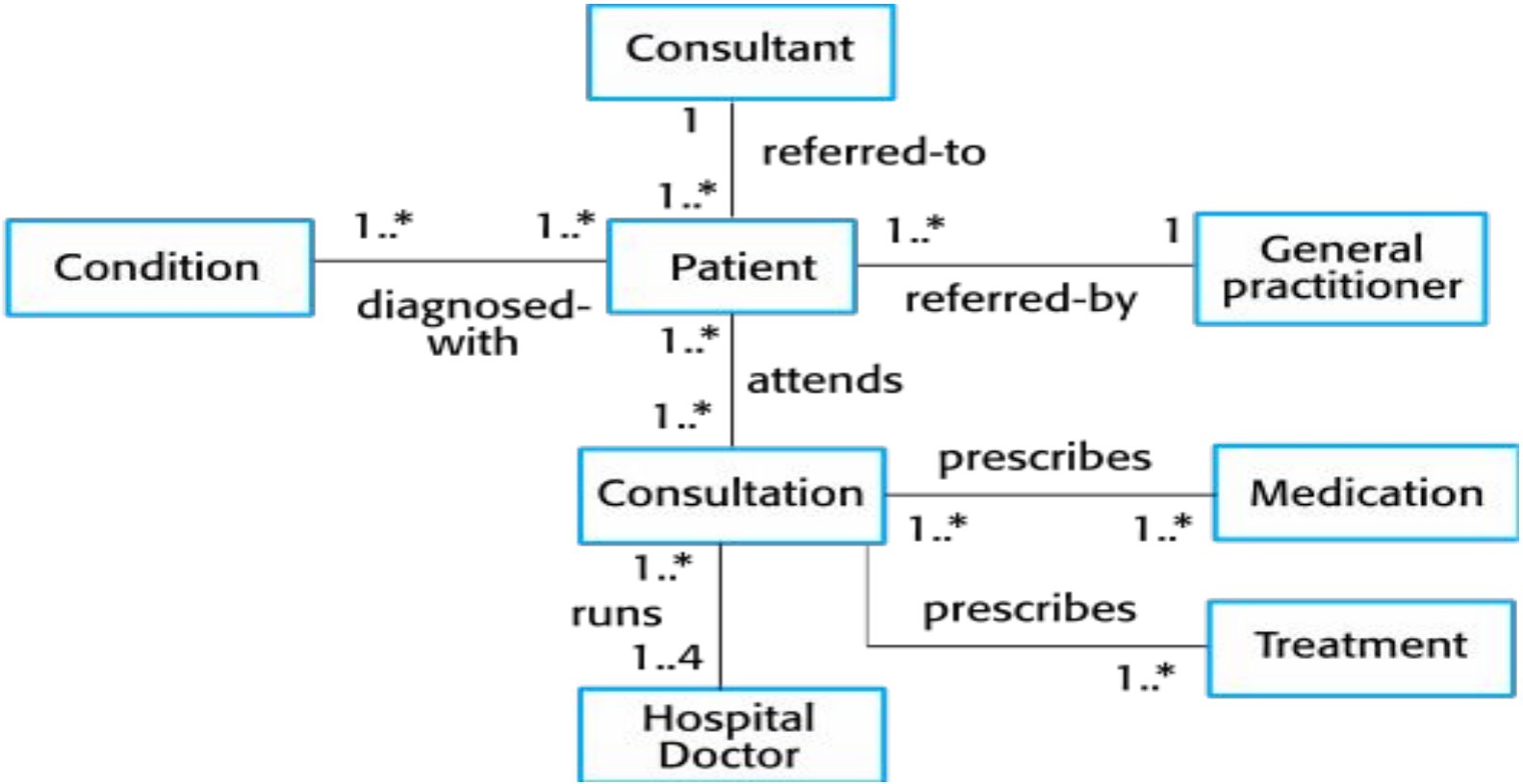
- Structural models of software display the organization of a system in terms of the components that make up that system and their relationships.
- Structural models may be static models, which show the structure of the system design, or dynamic models, which show the organization of the system when it is executing.
- You create structural models of a system when you are discussing and designing the system architecture.

Class diagrams

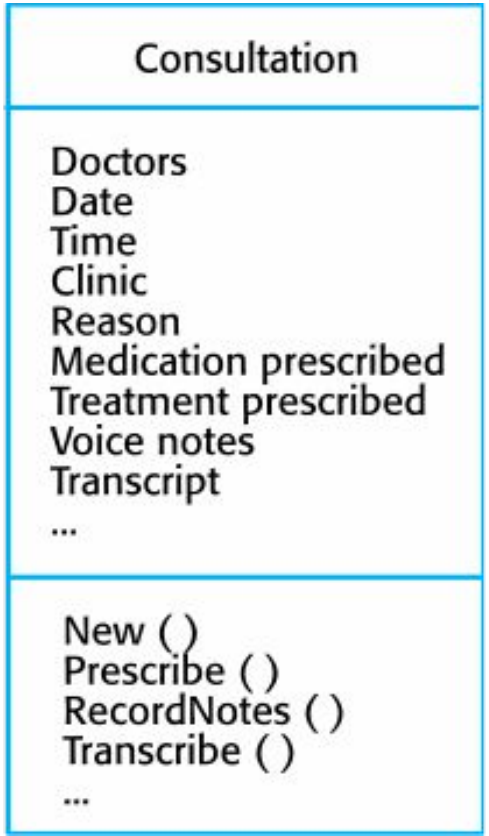
- Class diagrams are used when developing an object-oriented system model to show the classes in a system and the associations between these classes.
- An object class can be thought of as a general definition of one kind of system object.
- An association is a link between classes that indicates that there is some relationship between these classes.
- When you are developing models during the early stages of the software engineering process, objects represent something in the real world, such as a patient, a prescription, doctor, etc.

UML Classes and Association

Classes and Associations in the MHC-PMS



The consultation class



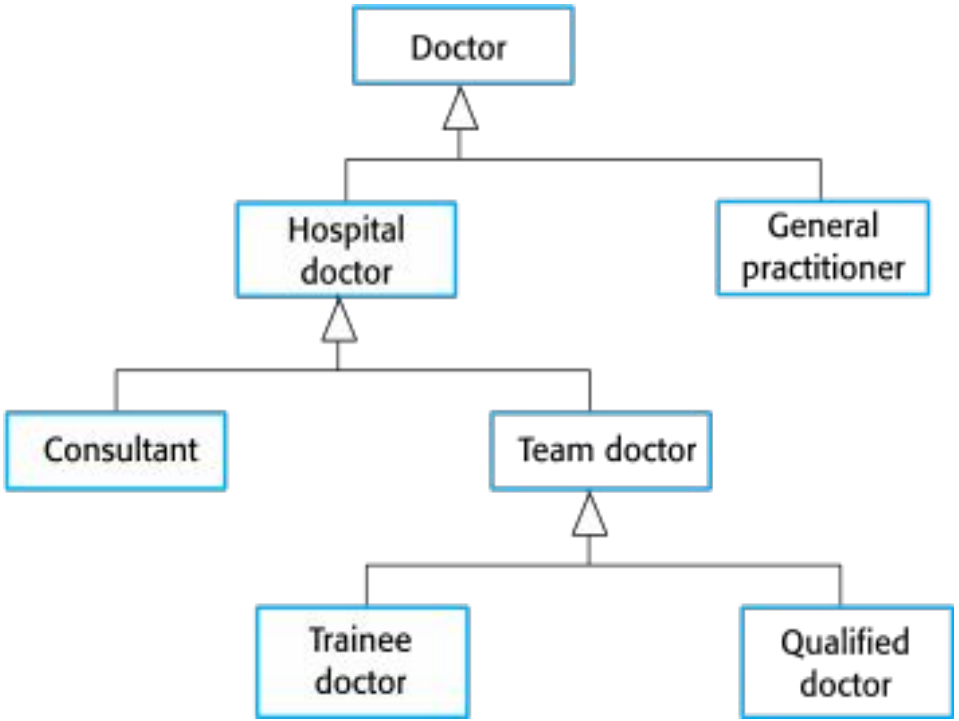
Generalization(1 of 2)

- Generalization is an everyday technique that we use to manage complexity.
- Rather than learn the detailed characteristics of every entity that we experience, we place these entities in more general classes (animals, cars, houses, etc.) and learn the characteristics of these classes.
- This allows us to infer that different members of these classes have some common characteristics e.g. squirrels and rats are rodents.

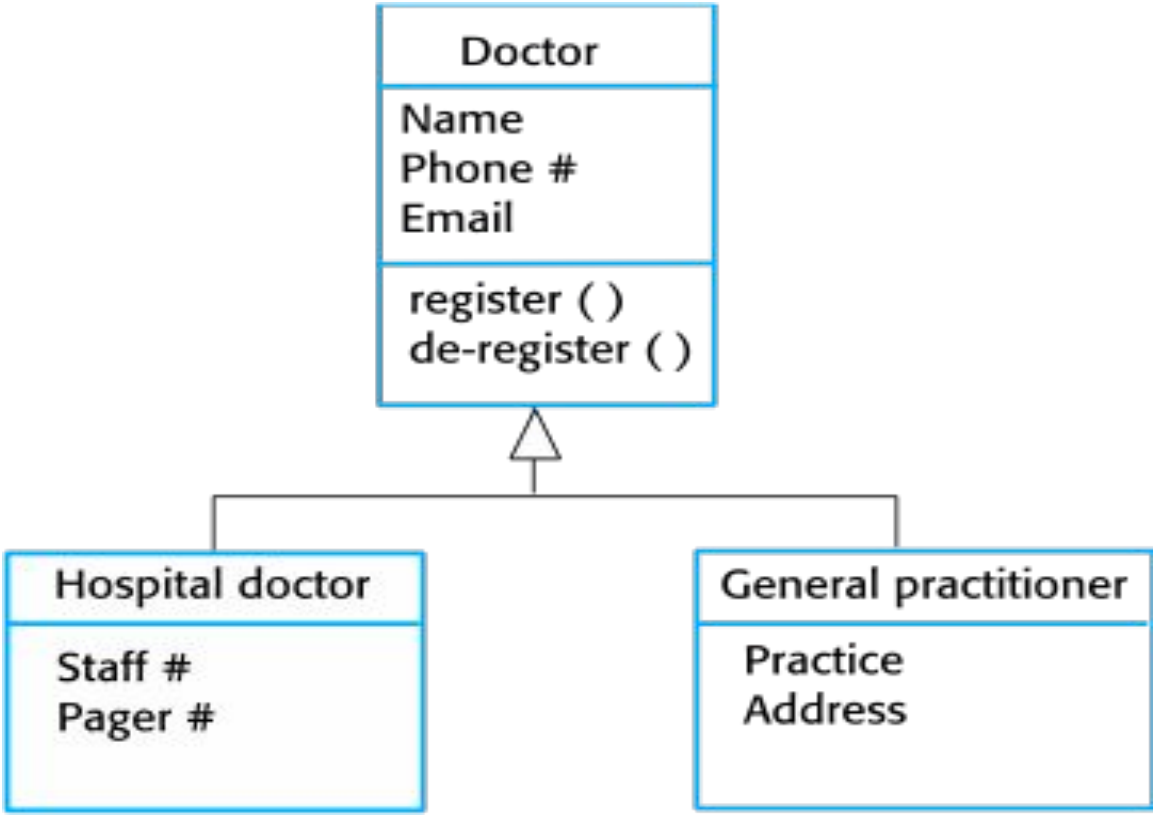
Generalization

- In modeling systems, it is often useful to examine the classes in a system to see if there is scope for generalization. If changes are proposed, then you do not have to look at all classes in the system to see if they are affected by the change.
- In object-oriented languages, such as Java, generalization is implemented using the class inheritance mechanisms built into the language.
- In a generalization, the attributes and operations associated with higher-level classes are also associated with the lower-level classes.
- The lower-level classes are subclasses inherit the attributes and operations from their superclasses. These lower-level classes then add more specific attributes and operations.

A Generalization Hierarchy



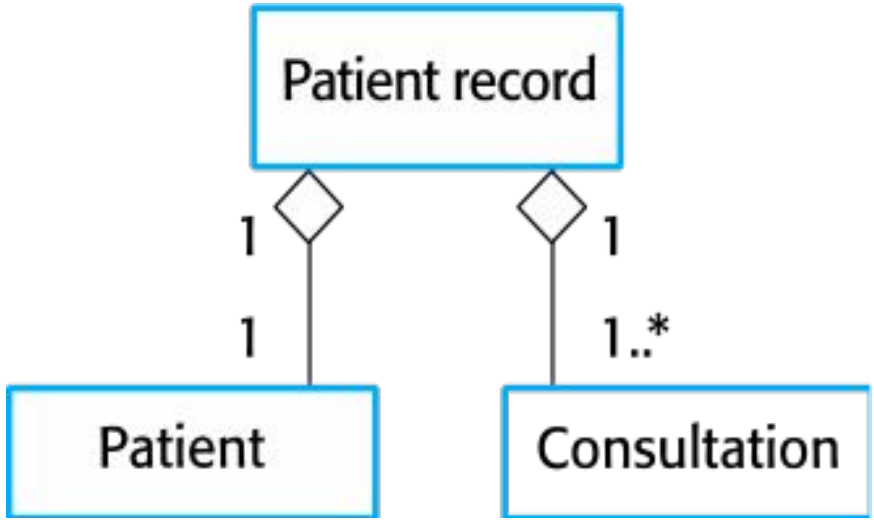
Generalization Hierarchy with added detail



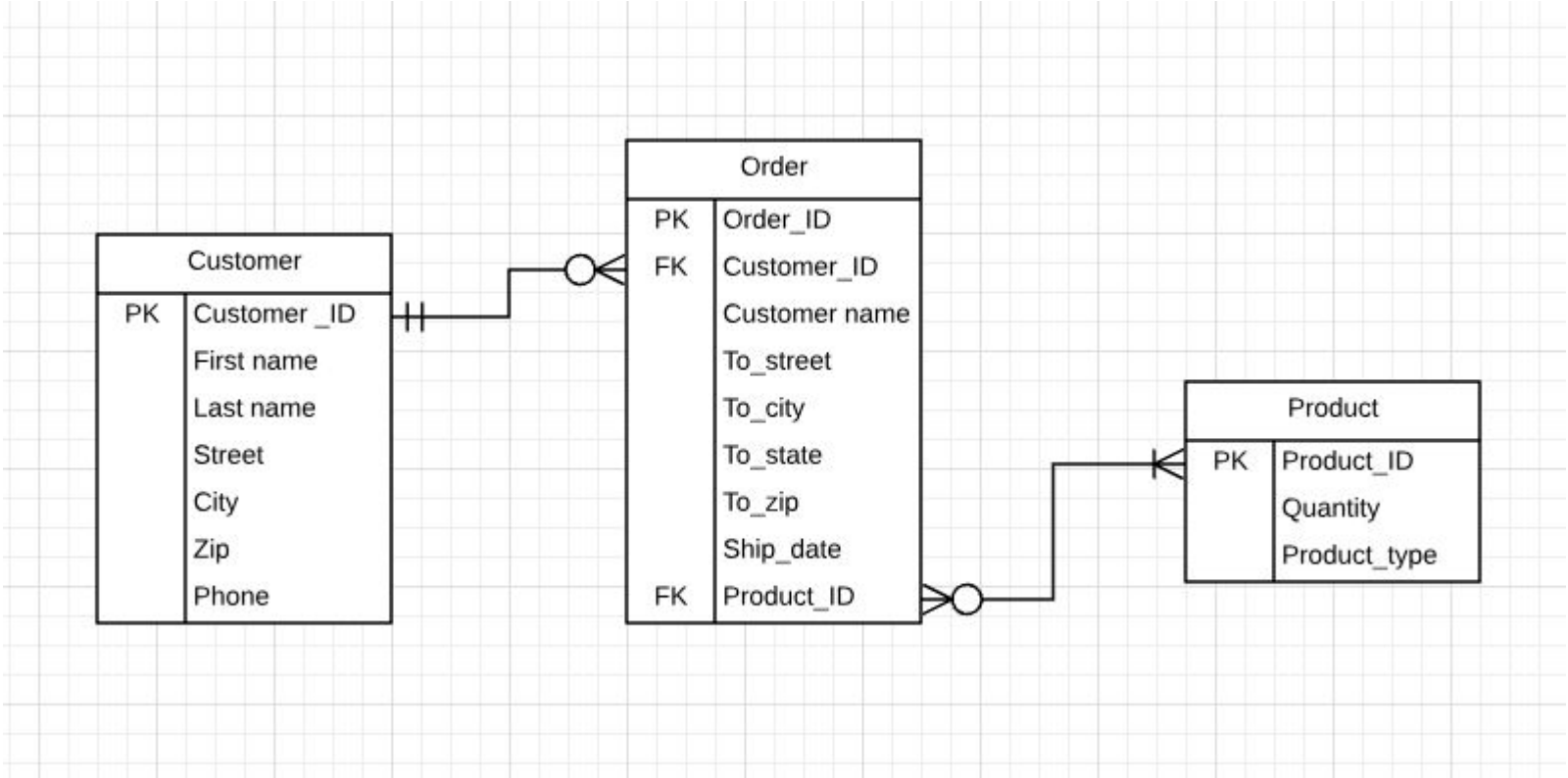
Object Class Aggregation Models

- An aggregation model shows how classes that are collections are composed of other classes.
- Aggregation models are similar to the part-of relationship in semantic data models.

The Aggregation Association



Entity Relationship Diagram



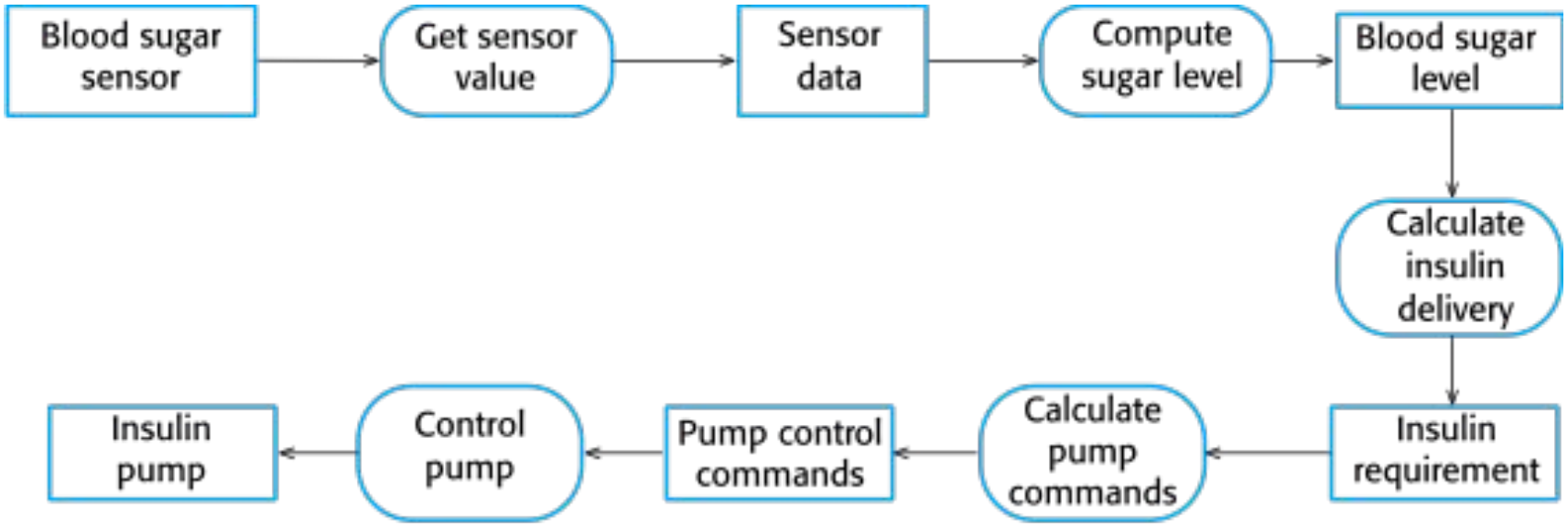
Behavioral Models

- Behavioral models are models of the dynamic behavior of a system as it is executing. They show what happens or what is supposed to happen when a system responds to a stimulus from its environment.
- You can think of these stimuli as being of two types:
  - Data
    - Some data arrives that has to be processed by the system.
  - Events
    - Some event happens that triggers system processing. Events may have associated data, although this is not always the case.

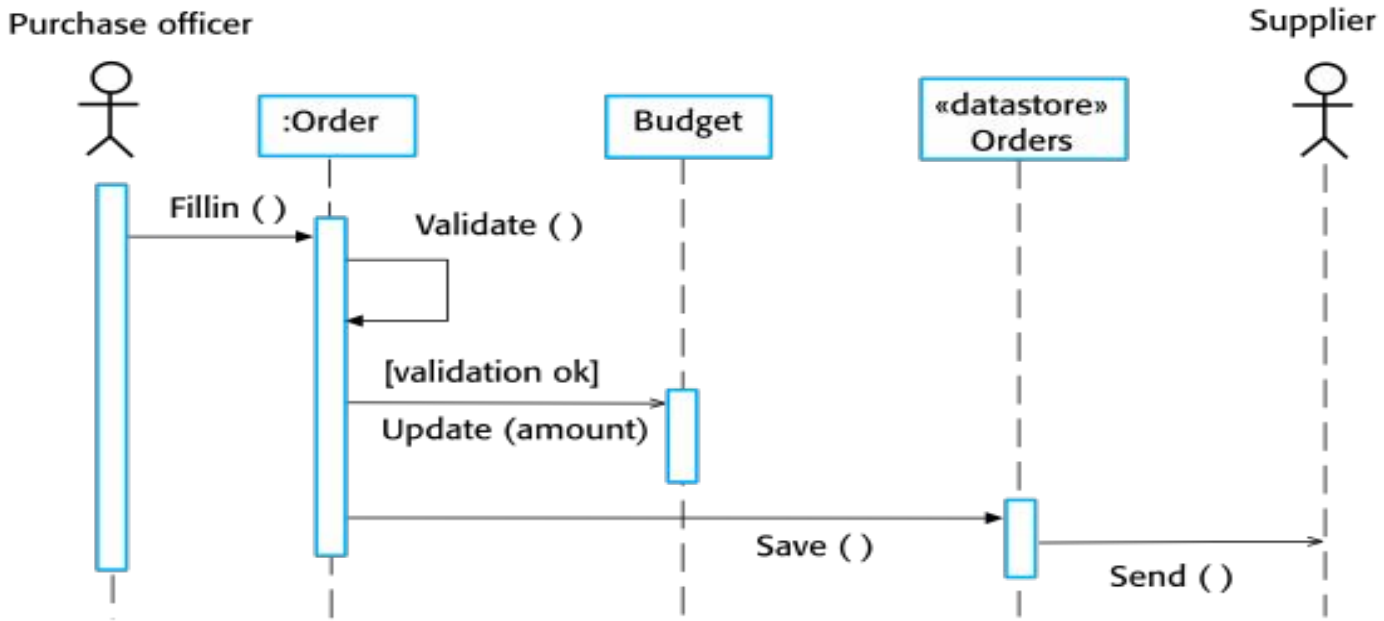
Data-Driven Modeling

- Many business systems are data-processing systems that are primarily driven by data. They are controlled by the data input to the system, with relatively little external event processing.
- Data-driven models show the sequence of actions involved in processing input data and generating an associated output.
- They are particularly useful during the analysis of requirements as they can be used to show end-to-end processing in a system.

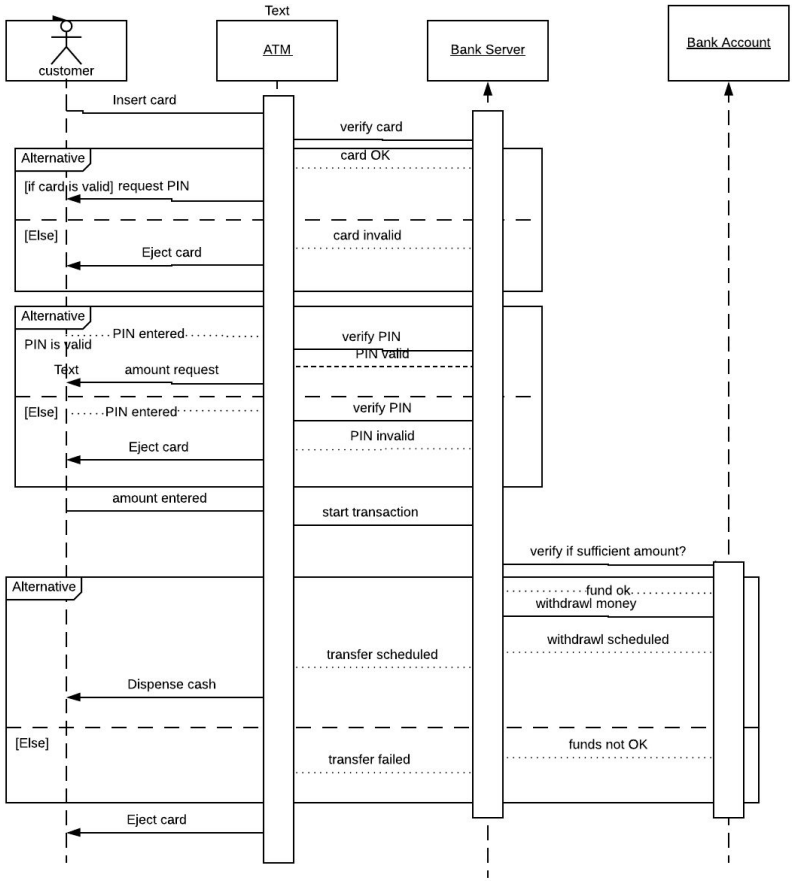
Activity diagram of the Insulin Pump's Operation



Sequence Diagram of Order Processing



Sequence diagram of ATM Cash Dispense



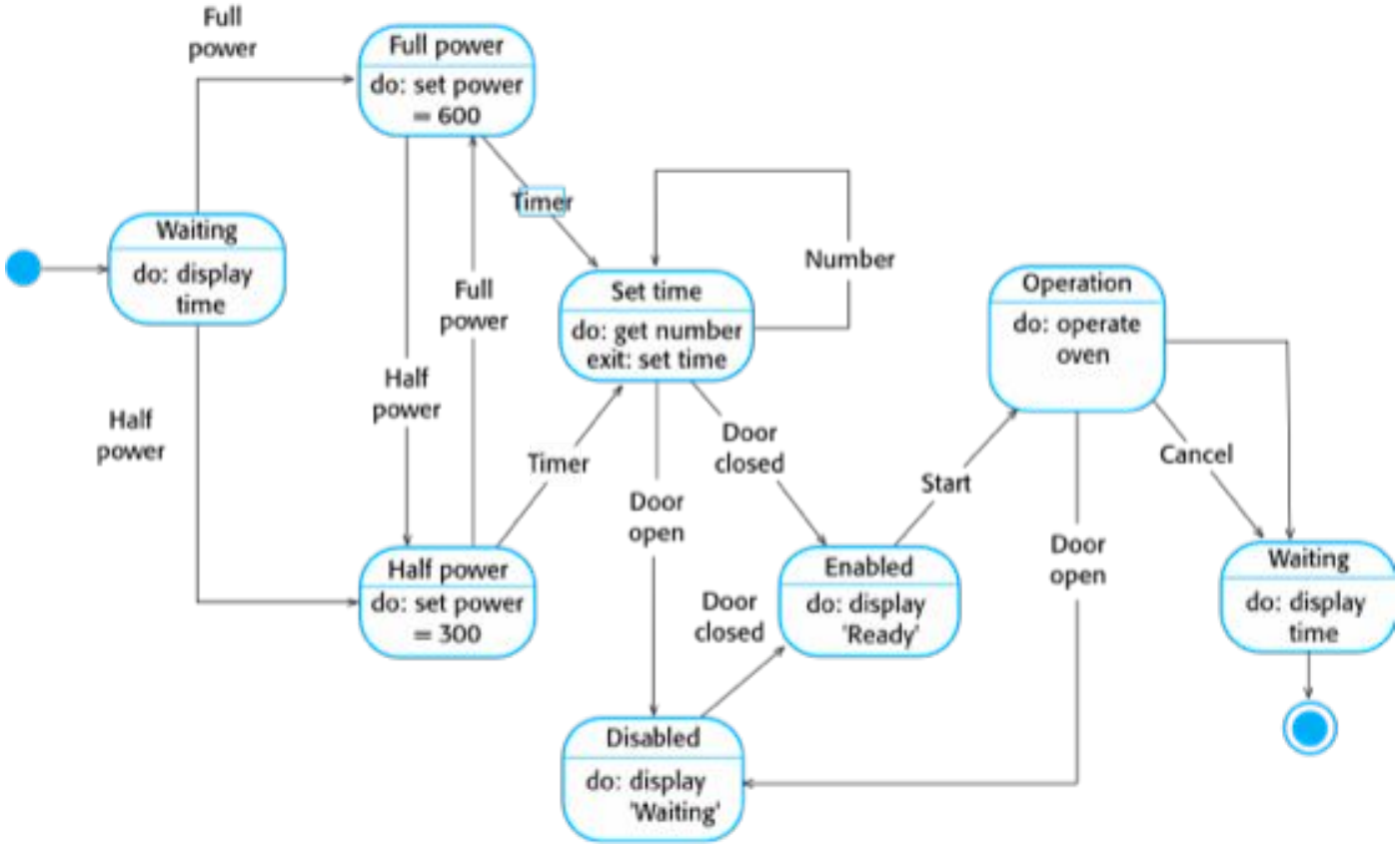
Event-Driven Modeling

- Real-time systems are often event-driven, with minimal data processing. For example, a landline phone switching system responds to events such as ‘receiver off hook’ by generating a dial tone.
- Event-driven modeling shows how a system responds to external and internal events.
- It is based on the assumption that a system has a finite number of states and that events (stimuli) may cause a transition from one state to another.

State Machine Models

- These model the behaviour of the system in response to external and internal events.
- They show the system's responses to stimuli so are often used for modelling real-time systems.
- State machine models show system states as nodes and events as arcs between these nodes. When an event occurs, the system moves from one state to another.
- State charts are an integral part of the UML and are used to represent state machine models.

State Diagram of a Microwave Oven



State Diagram of Operation function