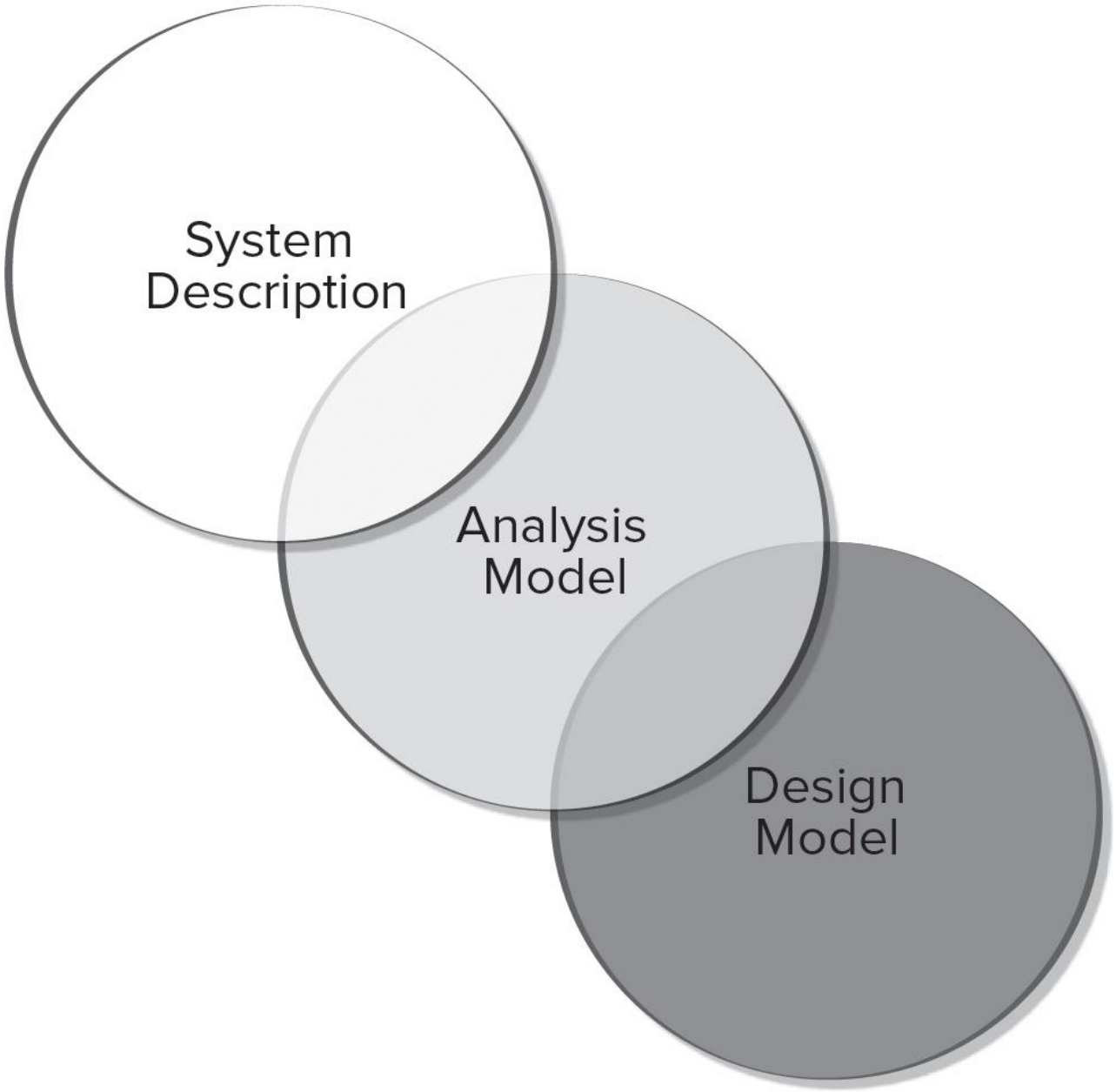
System Modeling:
From Requirements to Design

Prof. K. P. Hewagamage

Graphical Model Vs Textual Description

- The written word is a wonderful vehicle for communication, but it is not necessarily the best way to represent the requirements for computer software.
- At a technical level, software engineering begins with a series of modeling tasks that lead to a specification of requirements and a design representation for the software to be built.
- The requirements model is actually a set of models that make up the first technical representation of a system.

A Bridge



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.

Models

- ***Scenario-based models*** depict requirements from the point of view of various system "actors."
- ***Class-oriented models*** represent object-oriented classes (attributes and operations) and how classes collaborate to achieve system requirements.
- ***Behavioral models*** depict how the software reacts to internal or external "events."
- ***Data models*** depict the information domain for the problem.
- ***Flow-oriented models*** represent functional elements of the system and how they transform data in the system.

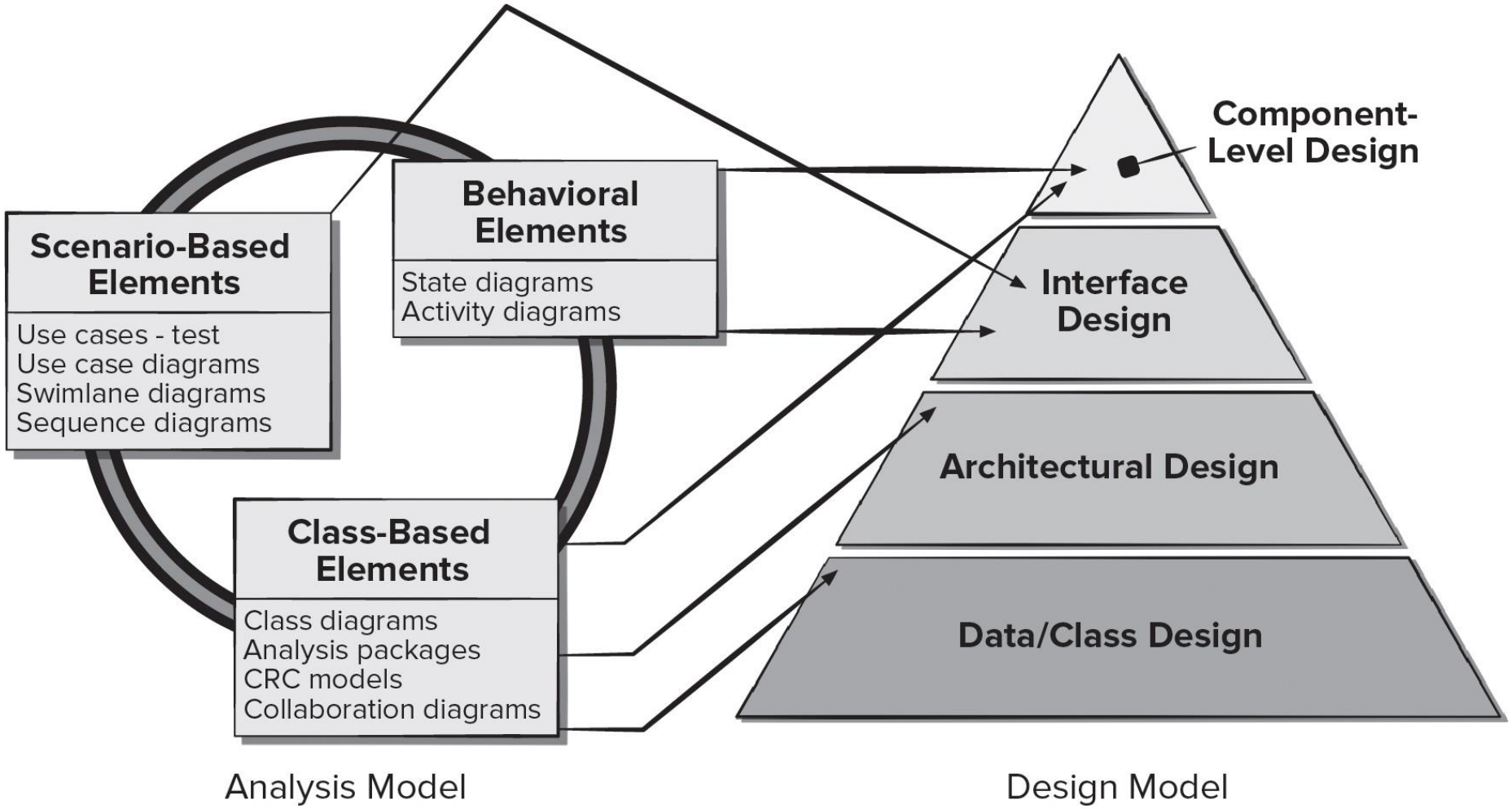
System Modeling

- System modeling is the process of developing abstract models of a system, with each model presenting a different view or perspective of that system.
- System modeling has now come to mean representing a system using some kind of graphical notation, which is now almost always based on notations in the Unified Modeling Language (UML).
- System modelling helps the analyst to understand the functionality of the system and models are used to communicate with customers.

Existing and planned system models

- Models of the existing system are used during requirements engineering. They help clarify what the existing system does and can be used as a basis for discussing its strengths and weaknesses. These then lead to requirements for the new system.
- Models of the new system are used during requirements engineering to help explain the proposed requirements to other system stakeholders. Engineers use these models to discuss design proposals and to document the system for implementation.
- In a model-driven engineering process, it is possible to generate a complete or partial system implementation from the system model.

Mapping Requirements Model to Design Model



Software Engineering Design

- Data/Class design – transforms analysis classes into implementation classes and data structures.
- Architectural design – defines relationships among the major software structural elements.
- Interface design – defines how software elements, hardware elements, and end-users communicate.
- Component-level design – transforms structural elements into procedural descriptions of software components.

System 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.

UML Diagram Types

- **Activity diagrams**, which show the activities involved in a process or in data processing .
- **Use case diagrams**, which show the interactions between a system and its environment.
- **Sequence diagrams**, which show interactions between actors and the system and between system components.
- **Class diagrams**, which show the object classes in the system and the associations between these classes.
- **State diagrams**, which show how the system reacts to internal and external events.

Introduction to Unified Modeling Language (UML)

- One of the most exciting and useful tools in the world of system development
- A visual modeling language
- Enables system builders to create blueprints that capture their visions in a standard, easy-to-understand way
- Provides a mechanism to effectively share and communicate these visions with others (Technical or Non technical)

Introduction to Unified Modeling Language (UML)

- It most directly unifies the methods of
  - Booch,
  - Rumbaugh (OMT) and
  - Jacobson

as well as the best ideas from a number of other methodologies.

UML – History

- The Unified Modelling Language is a standard graphical language for modelling object oriented software
  - At the end of the 1980s and the beginning of 1990s, the first object-oriented development processes appeared
  - The proliferation of methods and notations tended to cause considerable confusion
  - Two important methodologists **Rumbaugh and Booch** decided to merge their approaches in 1994.
    - They worked together at the Rational Software Corporation
  - In 1995, another methodologist, **Jacobson**, joined the team
    - His work focused on use cases
  - In 1997 the **Object Management Group (OMG)** started the process of UML standardization

Introduction to the UML

Unified Modeling Language (UML) – a set of modeling conventions that is used to specify or describe a software system in terms of objects.

- The UML does **not prescribe a method** for developing systems—**only a notation** that is now widely accepted as a standard for object modeling.

UML History

- The first public draft version – (version 0.8) – Oct 1995
- Feedback from public and Ivor Jacobson's inputs included – (ver. 0.9 Jul 1996, ver. 0.91 Oct 1996)
- Ver 1.0/1.1 was presented to OMG group for standardization in July/Sep 1997.
- Nov 1997 – UML adopted as the standard modelling language by OMG
- Ver 1.2 - June 1998
- Ver 1.3 – Dec 1998
- Ver 1.4 – 2000
- Ver 2.0 – 2003
- Ver 2.1 – 2007
- Ver 2.2 - 2009
- Ver 2.3 – 2010,
- Ver 2.4 –Jan 2011
- Ver 2.5 June 2015 .
- Ver 2.5.1 December 2017

UML Care Taker

- www.omg.org

- Object Management Group

- Watch Introduction - https://www.youtube.com/watch?v=vAHHdnIV8rU

UML Diagrams

- UML has 14 types of diagrams divided into two categories.

  - Structure Diagrams :

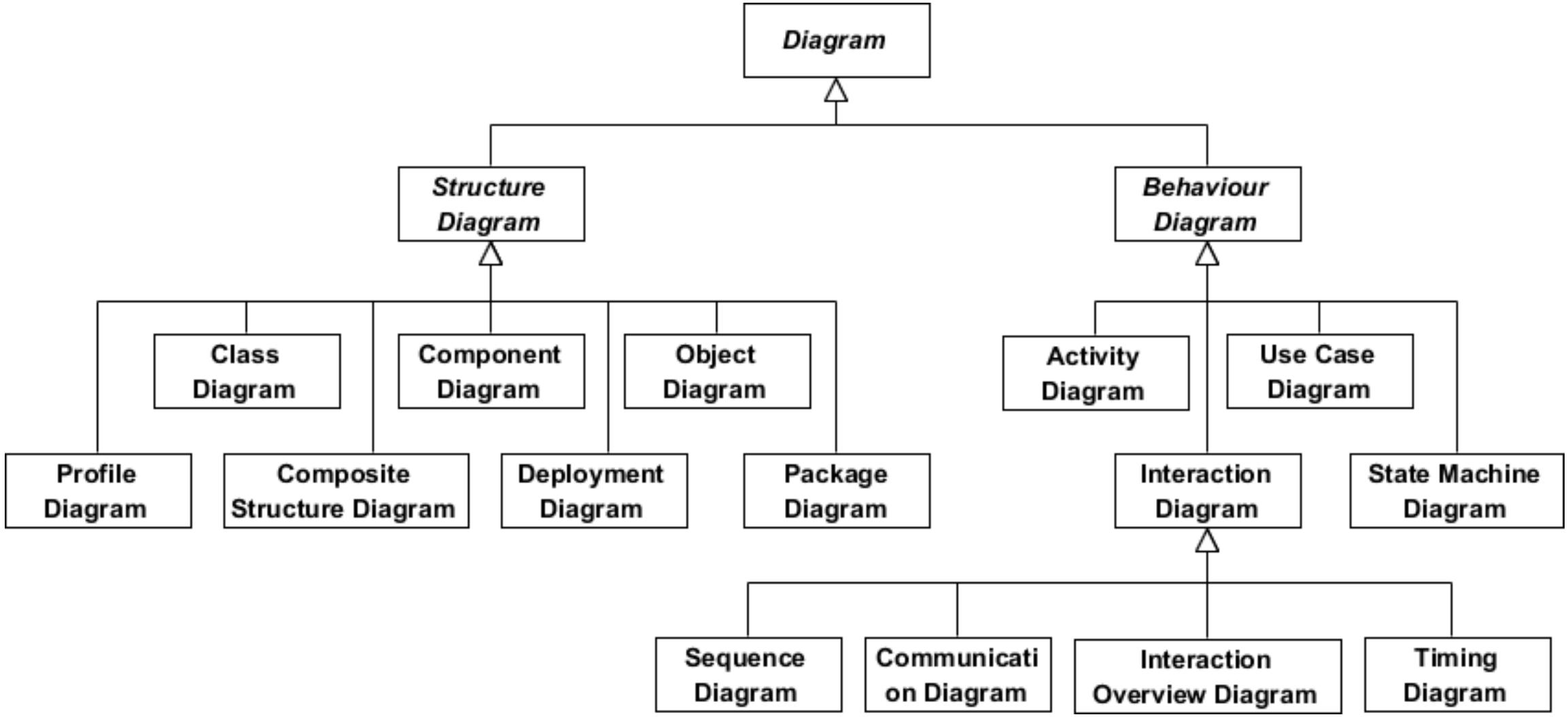
  Seven diagram types represent *structural* information,

  -Behaviour Diagrams :

  other seven represent general types of *behavior*..

The purpose of the diagrams is to present multiple views (model) of a system.

UML Diagrams



Ref. http://www.omg.org/spec/UML/2.5/PDF/

Key UML diagrams (2.0)

- Class diagrams
  - describe classes and their relationships
- Use Case diagrams
- Interaction diagrams
  - show the behaviour of systems in terms of how objects interact with each other
- State diagrams and activity diagrams
  - show how systems behave internally
- Component and deployment diagrams
  - show how the various components of systems are arranged logically and physically

Context models

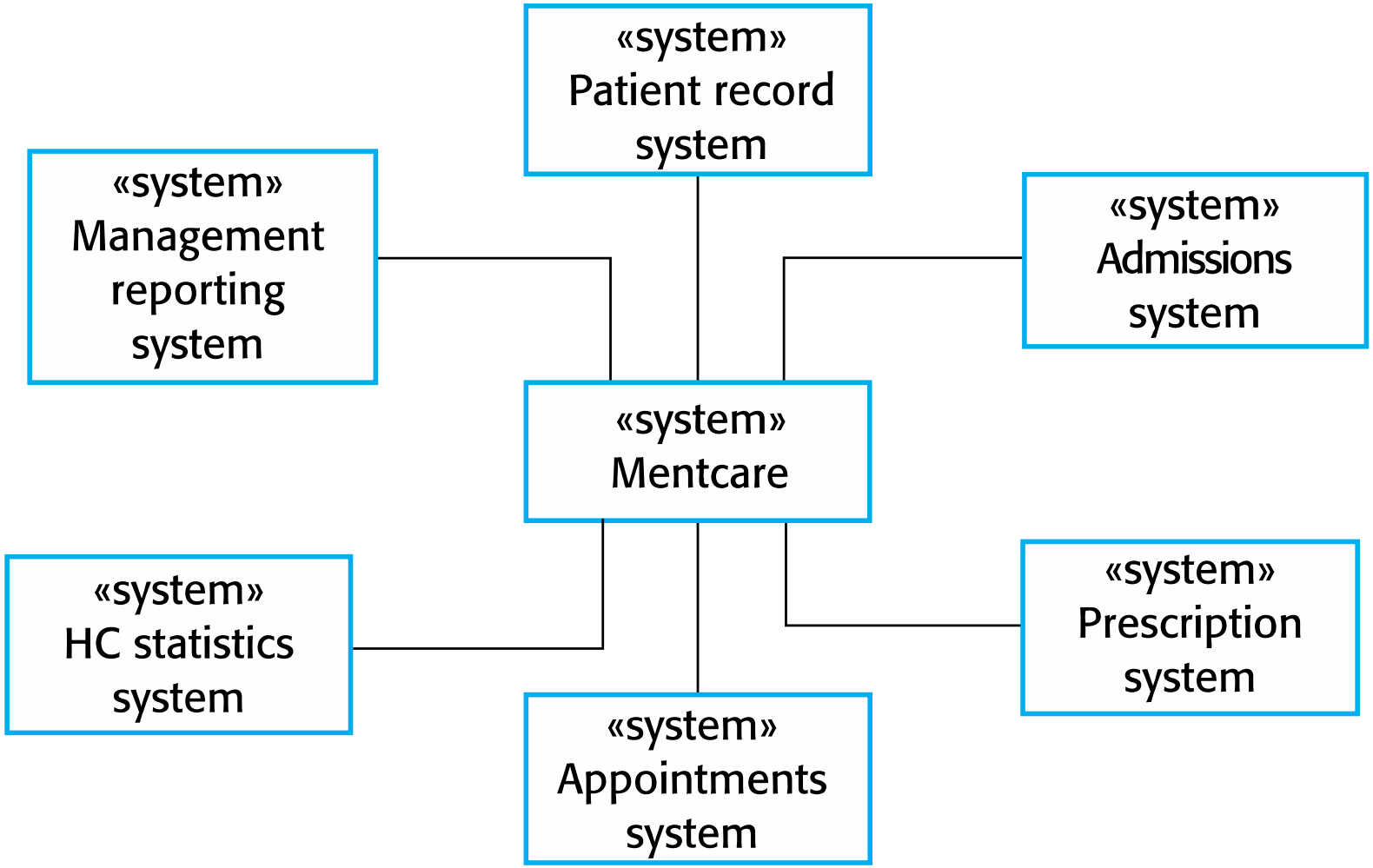
Context models

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

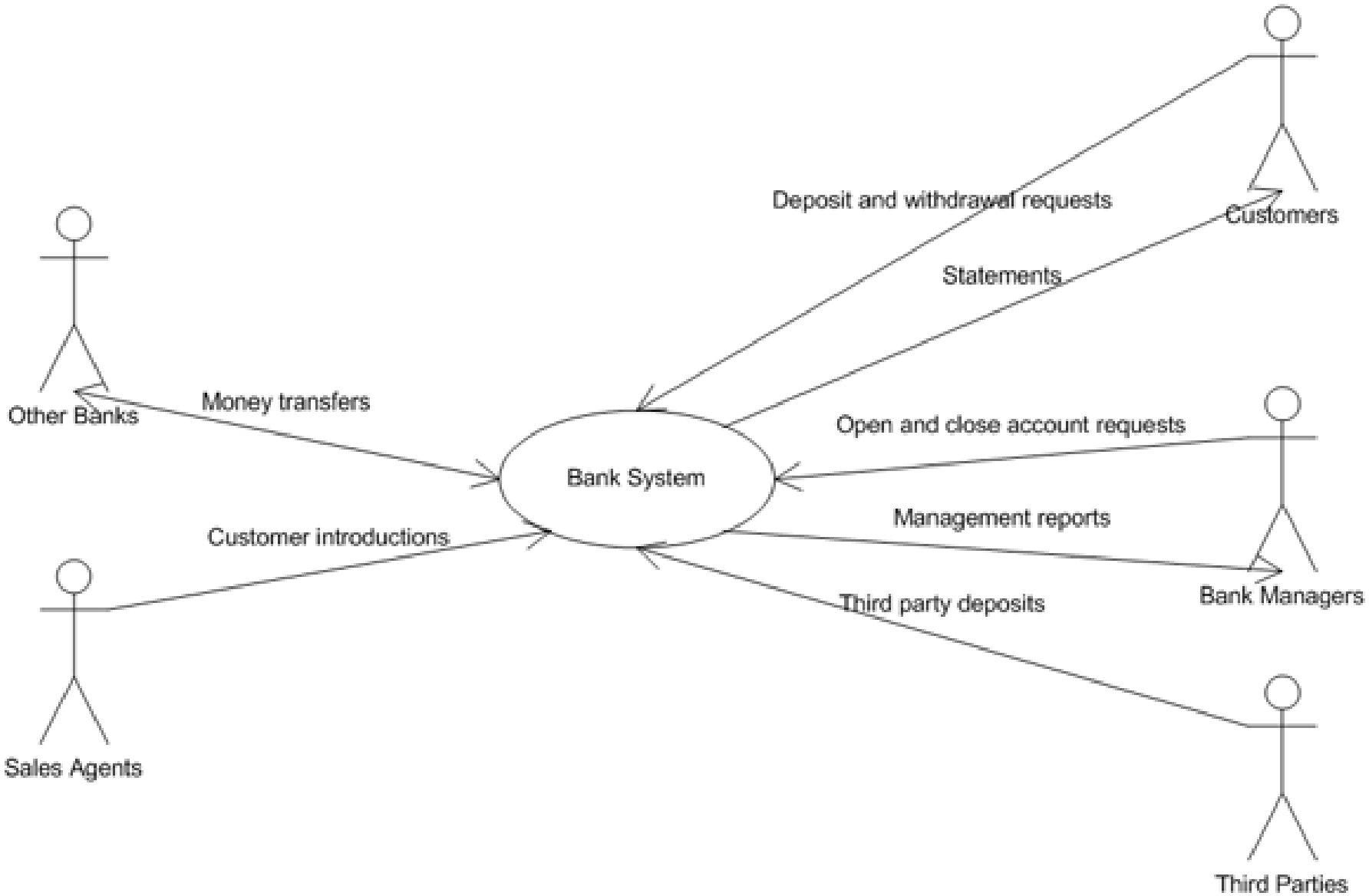
System boundaries

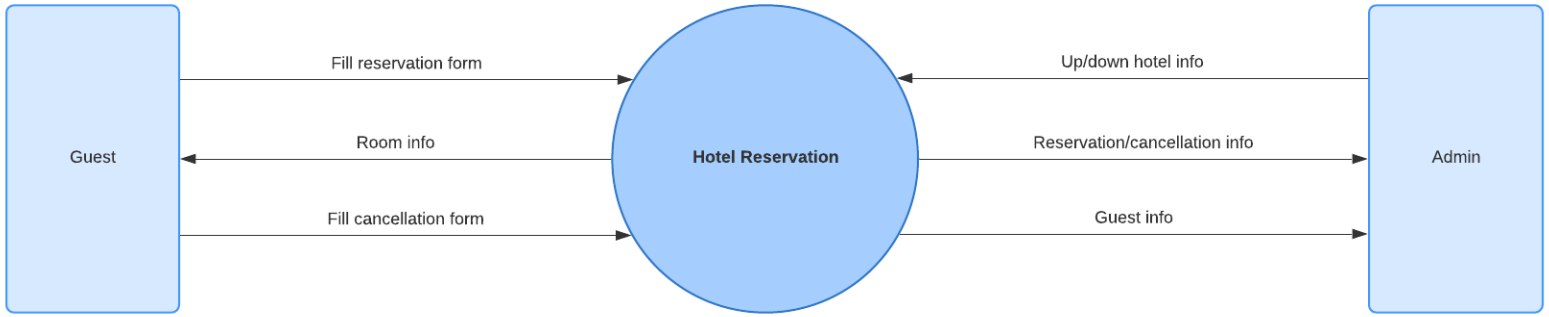
- System boundaries are established to define what is inside and what is outside the system.
  - They show other systems that are used or depend on the system being developed.
- The position of the system boundary has a profound effect on the system requirements.
- Defining a system boundary is a political judgment
  - There may be pressures to develop system boundaries that increase / decrease the influence or workload of different parts of an organization.

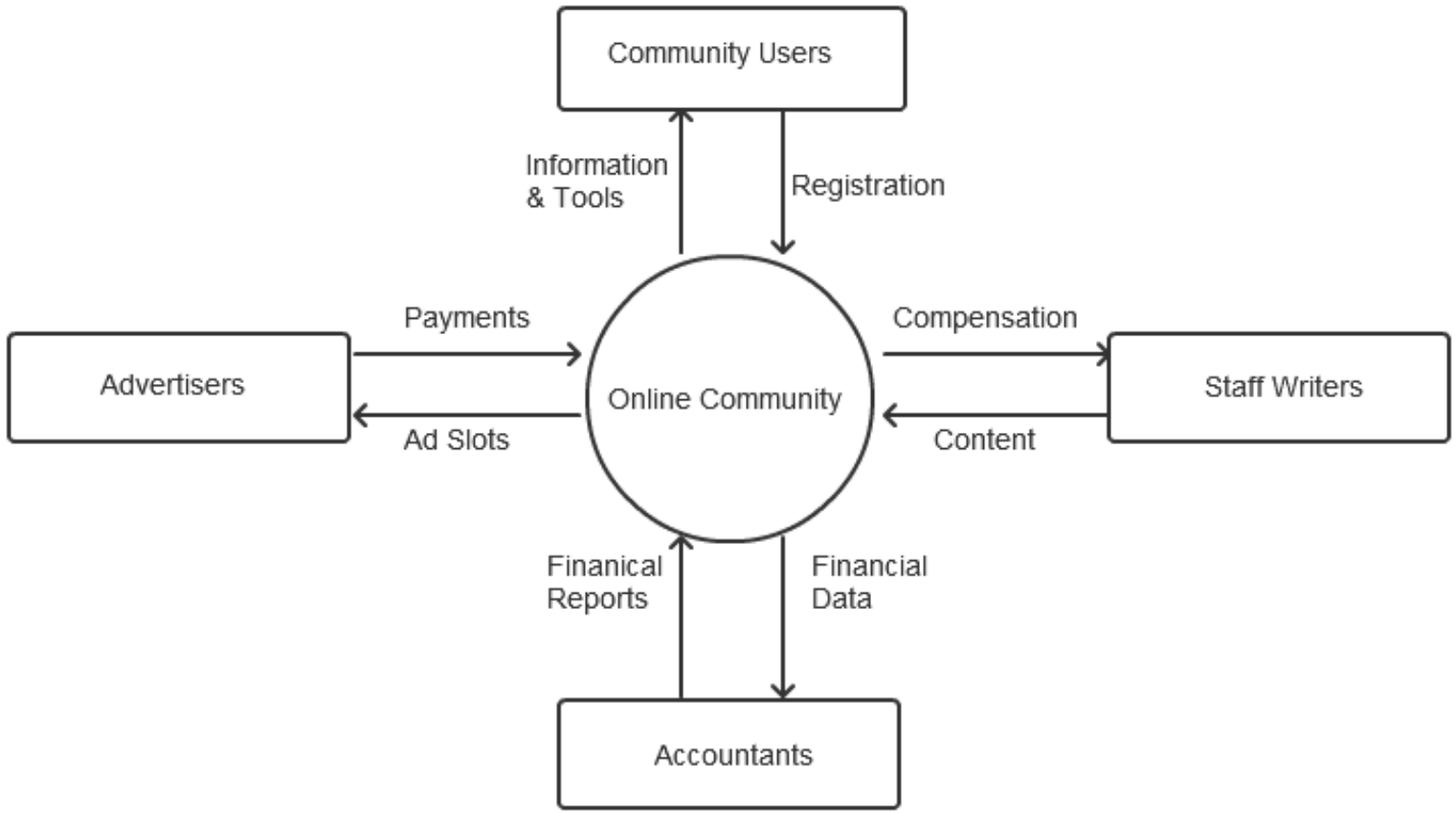
The context of the Mentcare system



Context Models



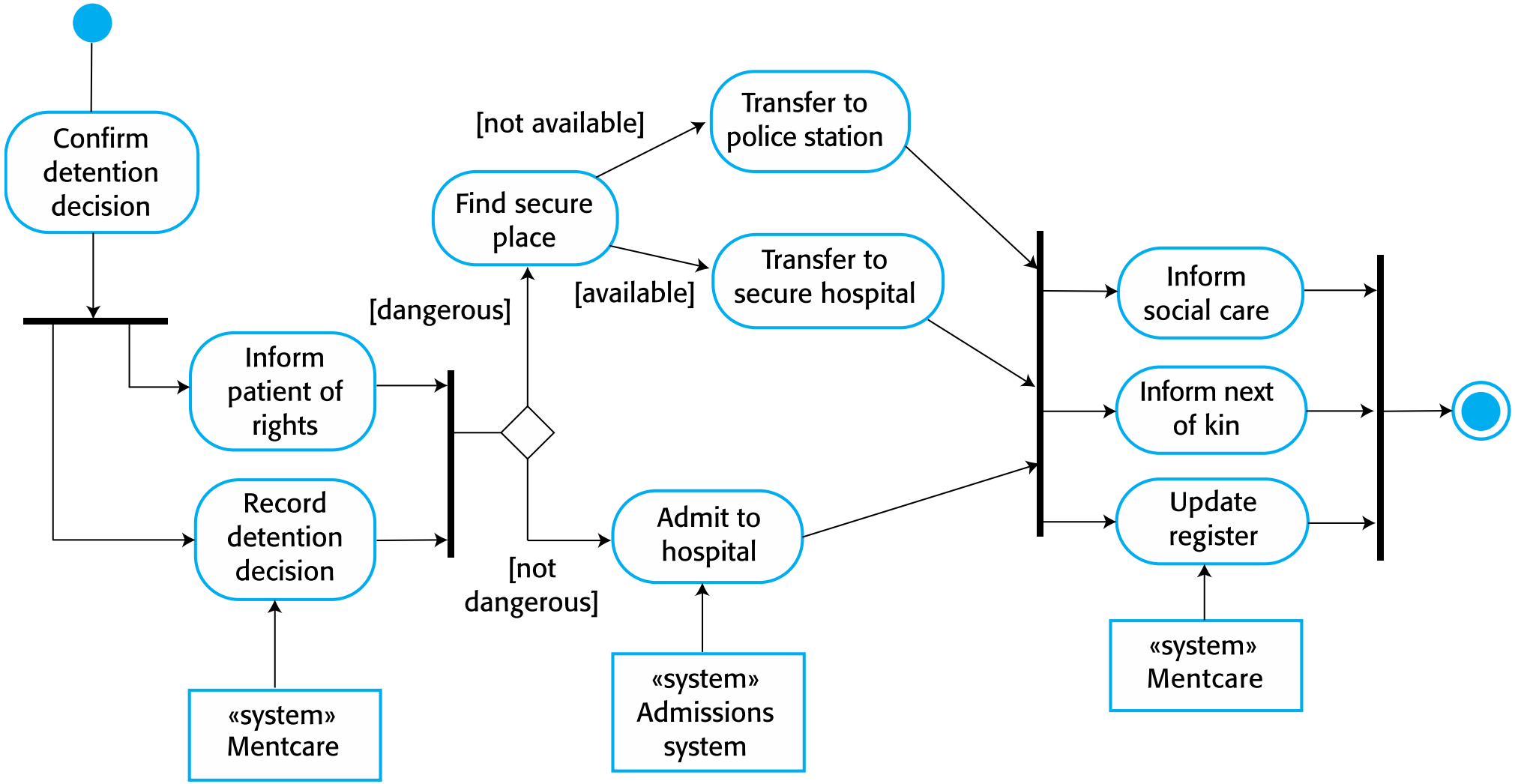

Guest
Fill reservation form
Room info
Fill cancellation form
Hotel Reservation
Up/down hotel info
Reservation/cancellation info
Guest info
Admin


Community Users
Information & Tools
Registration
Payments
Advertisers
Ad Slots
Online Community
Compensation
Staff Writers
Content
Finanical Reports
Financial Data
Accountants

Process Perspective

- Context models simply show the other systems in the environment, not how the system being developed is used in that environment.
- Process models reveal how the system being developed is used in broader business processes.
- UML activity diagrams may be used to define business process models.

Process model of involuntary detention



Interaction Models

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 case modeling

- Use cases were developed originally to support requirements elicitation and now incorporated into the UML.
- Each use case represents a discrete task that involves external interaction with a system.
- Actors in a use case may be people or other systems.
- Represented diagramatically to provide an overview of the use case and in a more detailed textual form.

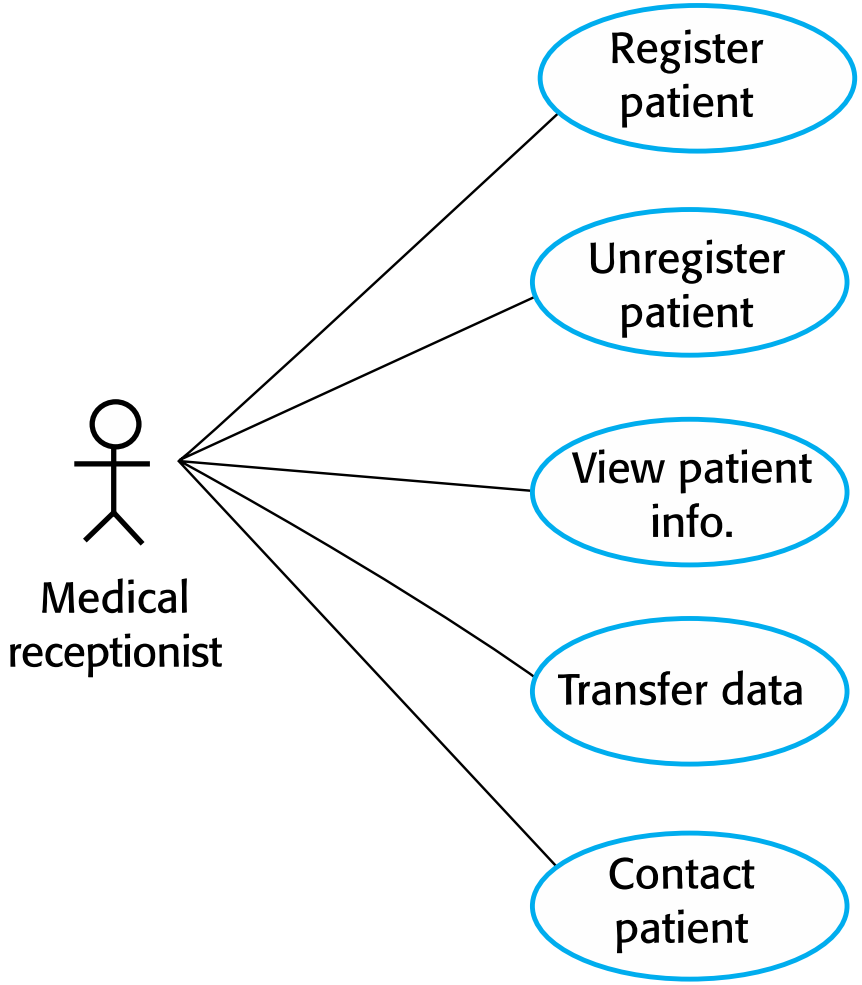
Transfer-data use case

- A use case in the Mentcare system

Tabular description of the ‘Transfer data’ use-case

MHC-PMS: Transfer data	
Actors	Medical receptionist, patient records system (PRS)
Description	A receptionist may transfer data from the Mentcase system to a general patient record database that is maintained by a health authority. The information transferred may either be updated personal information (address, phone number, etc.) or a summary of the patient’s diagnosis and treatment.
Data	Patient’s personal information, treatment summary
Stimulus	User command issued by medical receptionist
Response	Confirmation that PRS has been updated
Comments	The receptionist must have appropriate security permissions to access the patient information and the PRS.

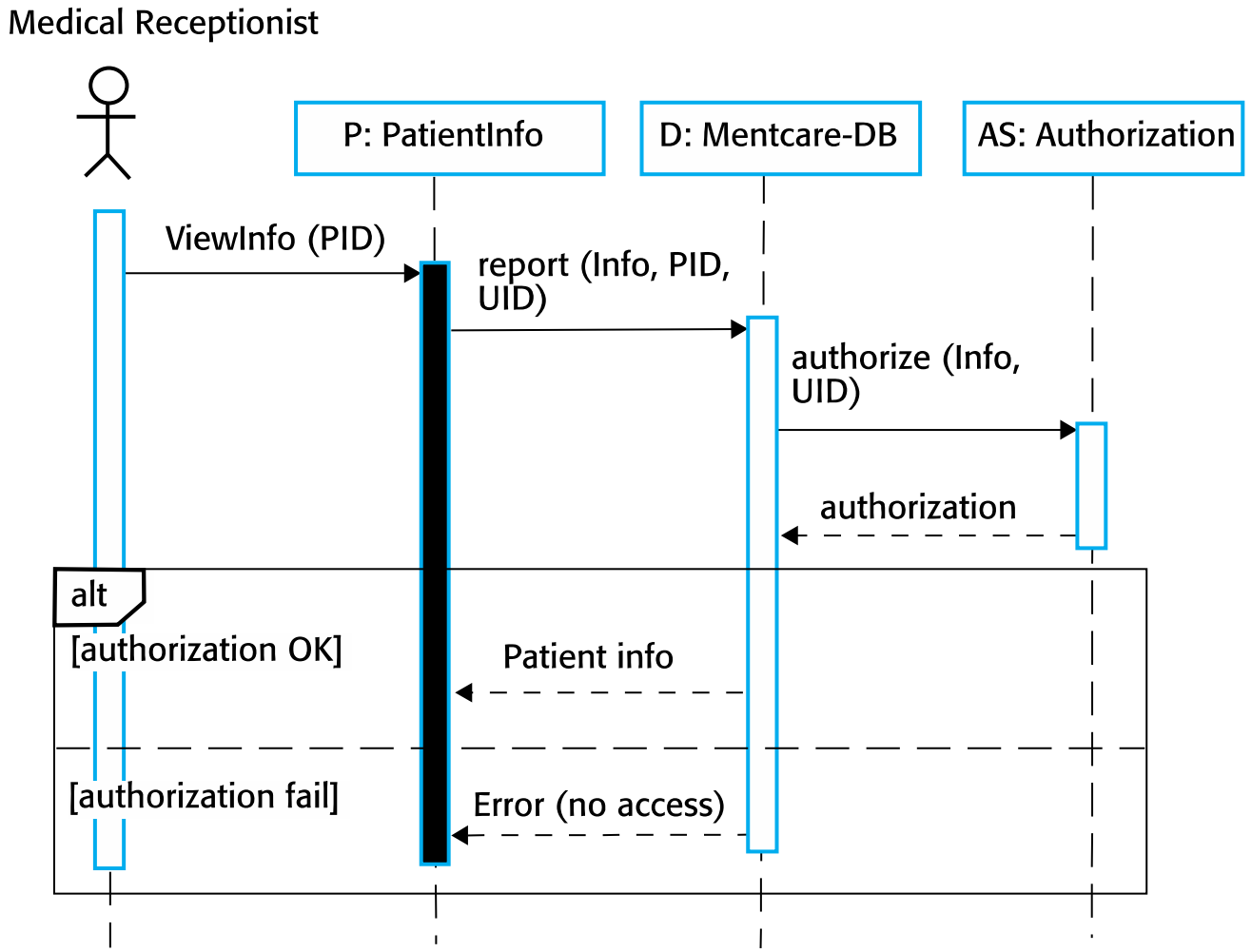
Use cases in the Mentcare system involving the role ‘Medical Receptionist’



Sequence Diagrams

- Sequence diagrams are part of the UML and are used to model the interactions between the actors and the objects within a system.
- A sequence diagram shows the sequence of interactions that take place during a particular use case or use case instance.
- The objects and actors involved are listed along the top of the diagram, with a dotted line drawn vertically from these.
- Interactions between objects are indicated by annotated arrows.

Sequence diagram for View patient information



Structural Models

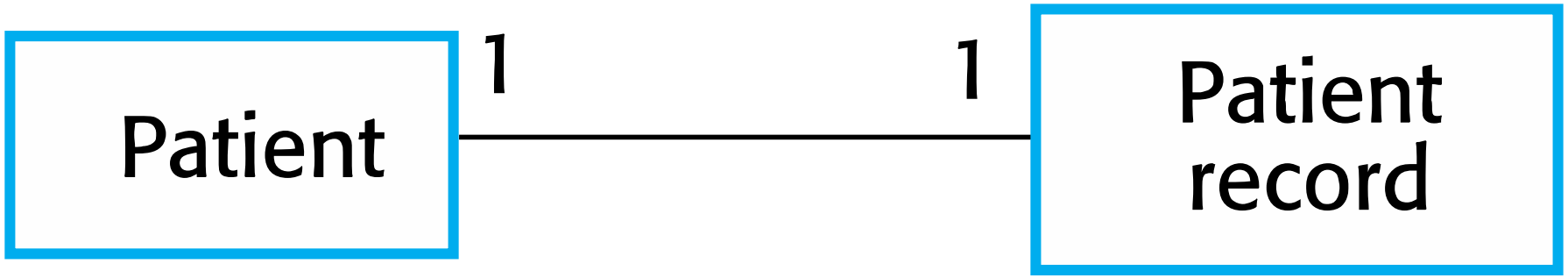
Structural Models

- 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.

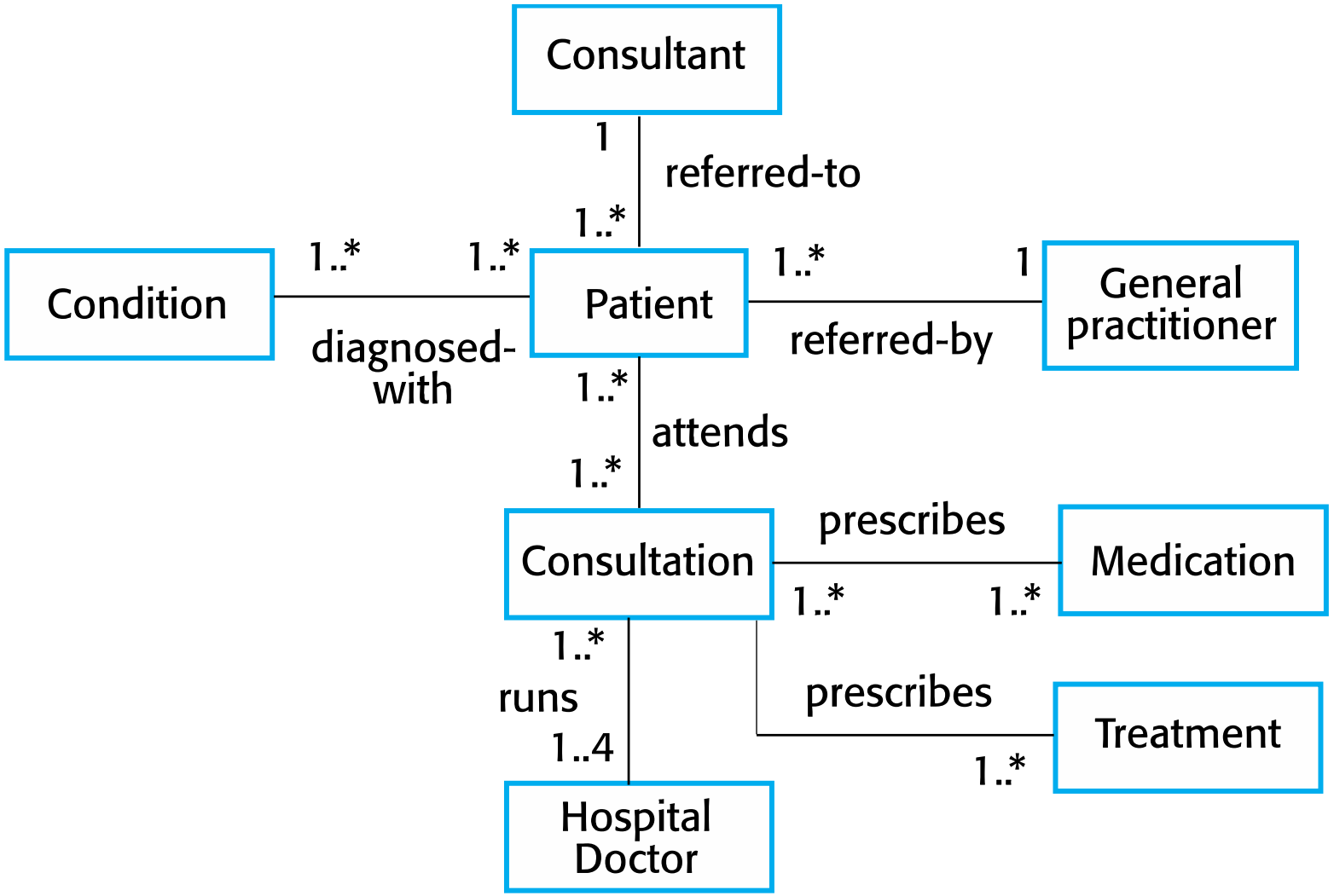
Class Diagrams

- 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.

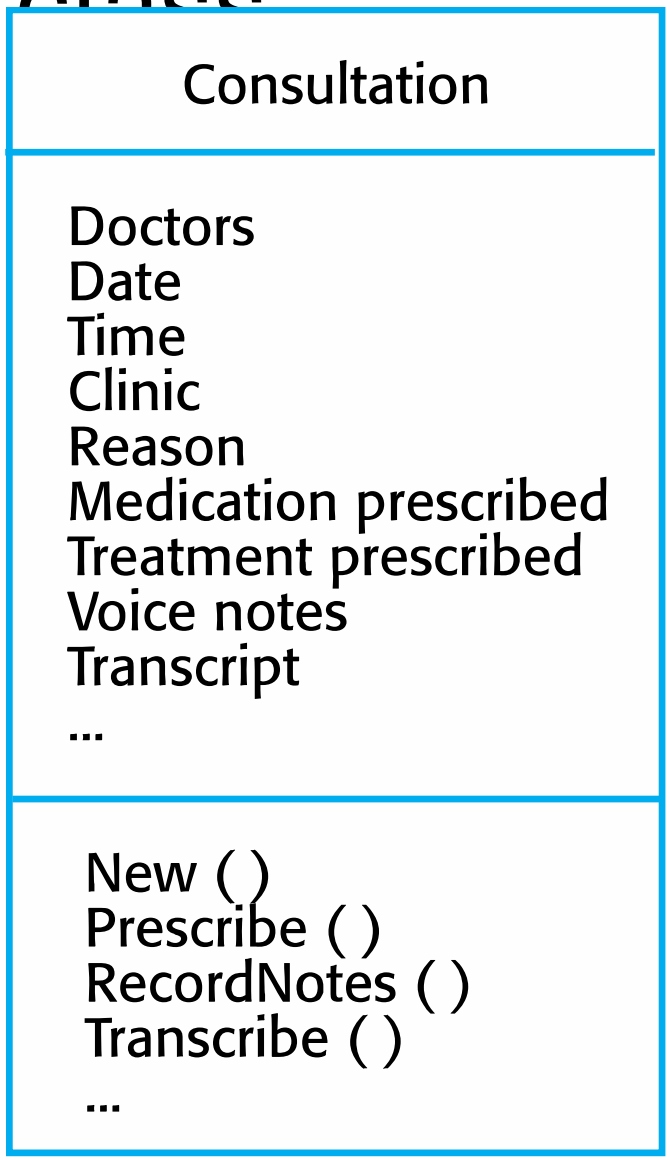
UML classes and association



Classes and associations in the MHC-PMS



The Consultation class



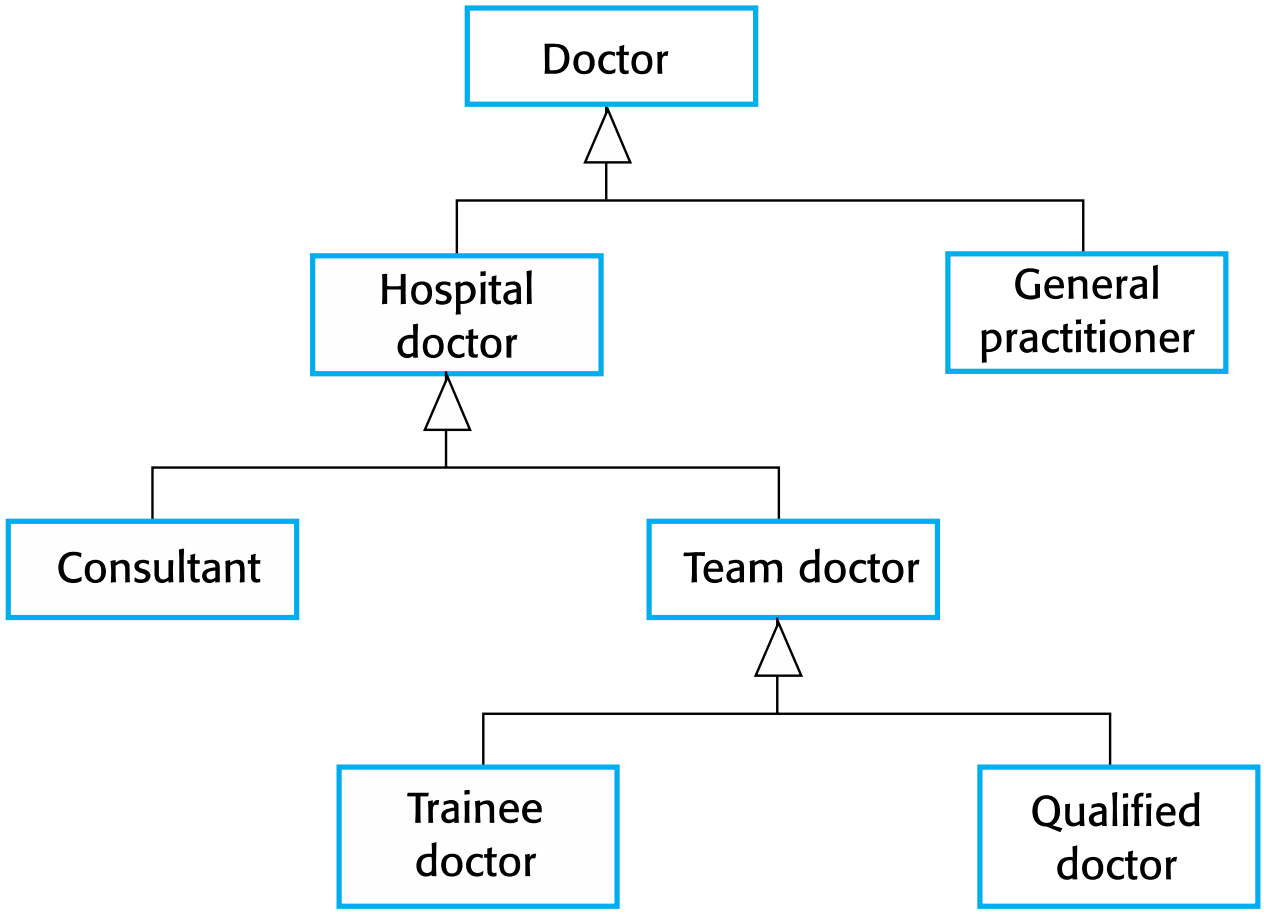
Generalization

- 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.

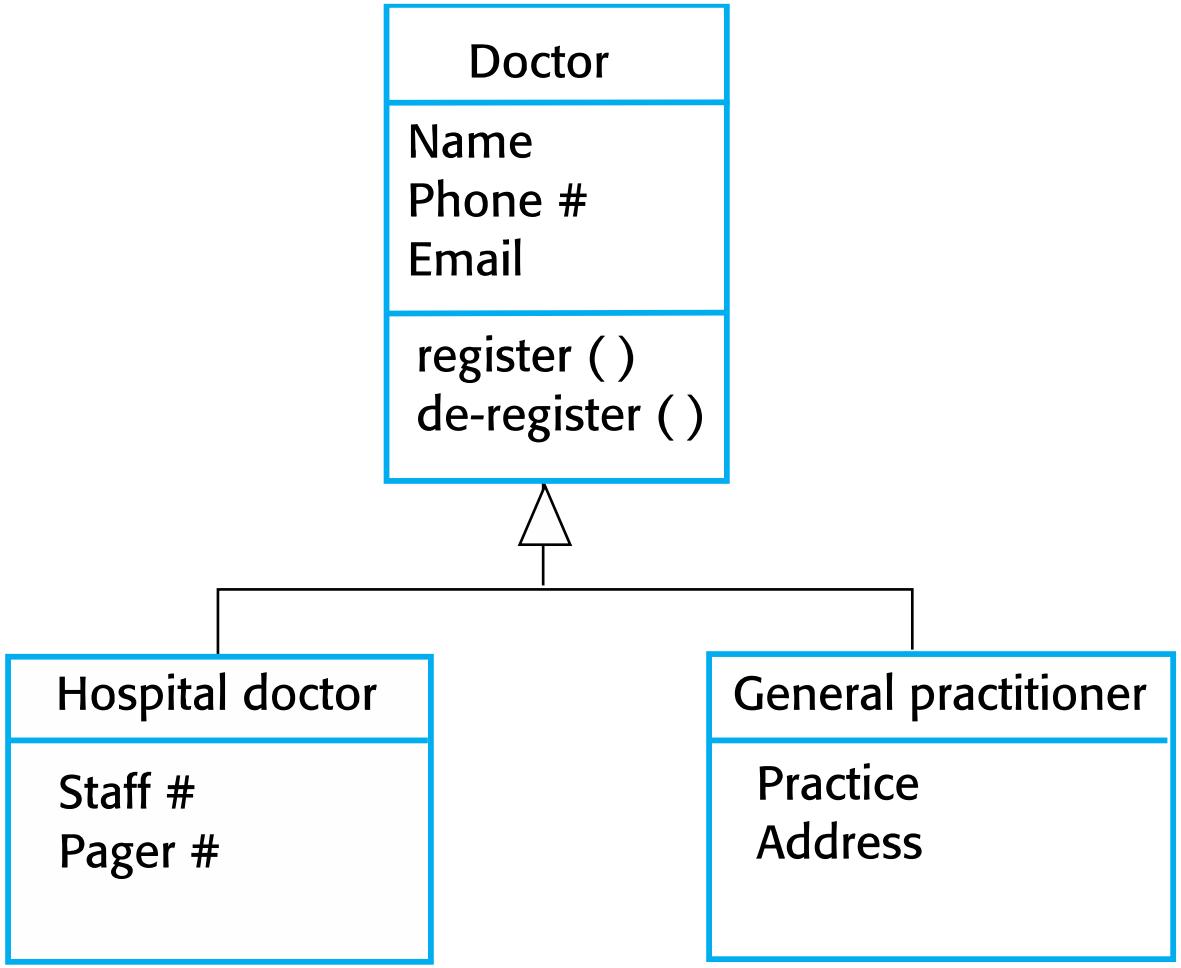
Generalization ....

- 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



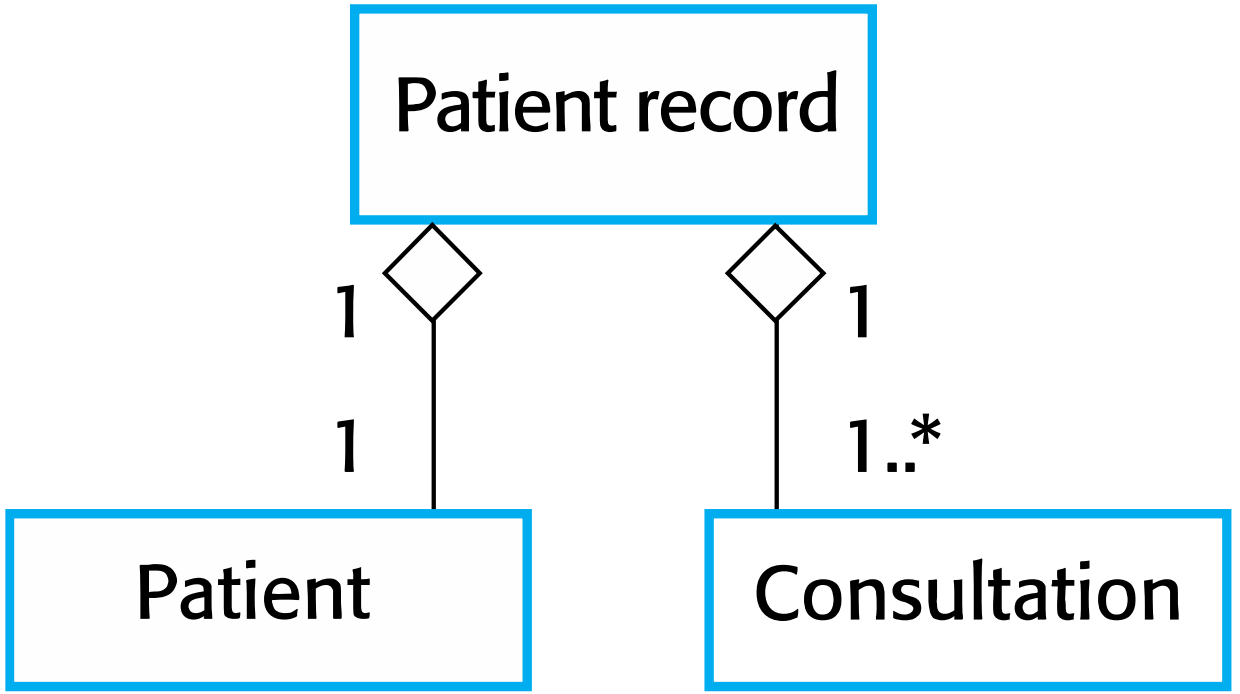
A generalization hierarchy with added detail



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



Behavioral Models

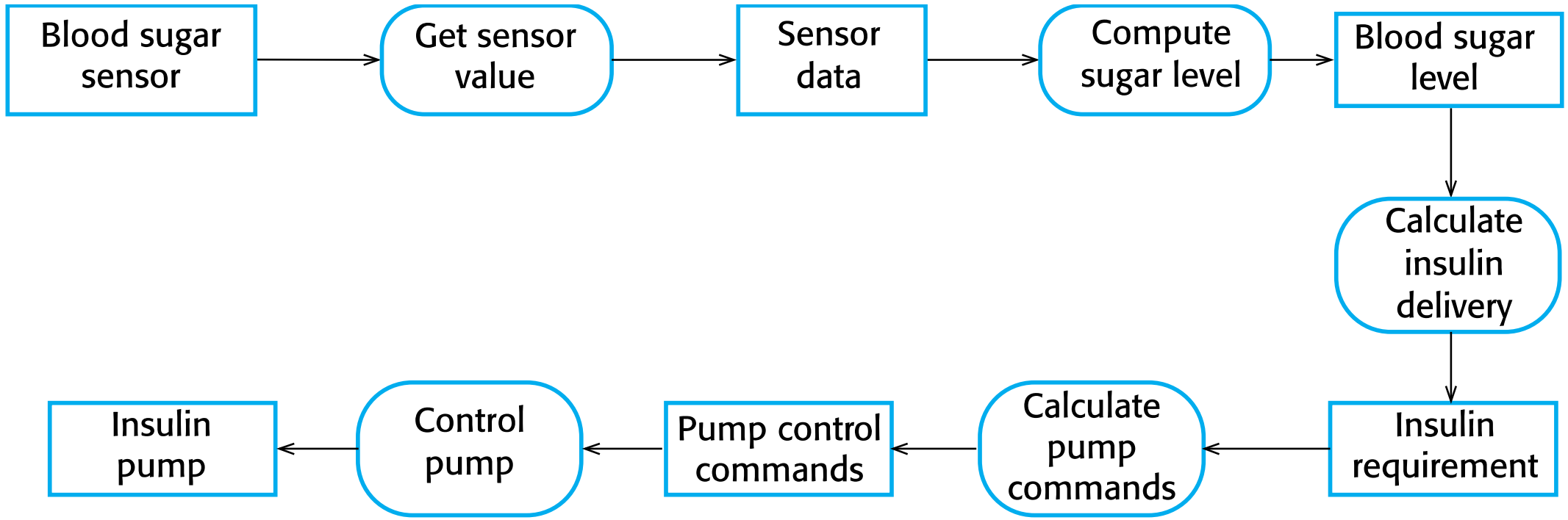
Behavioral Models

- 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.

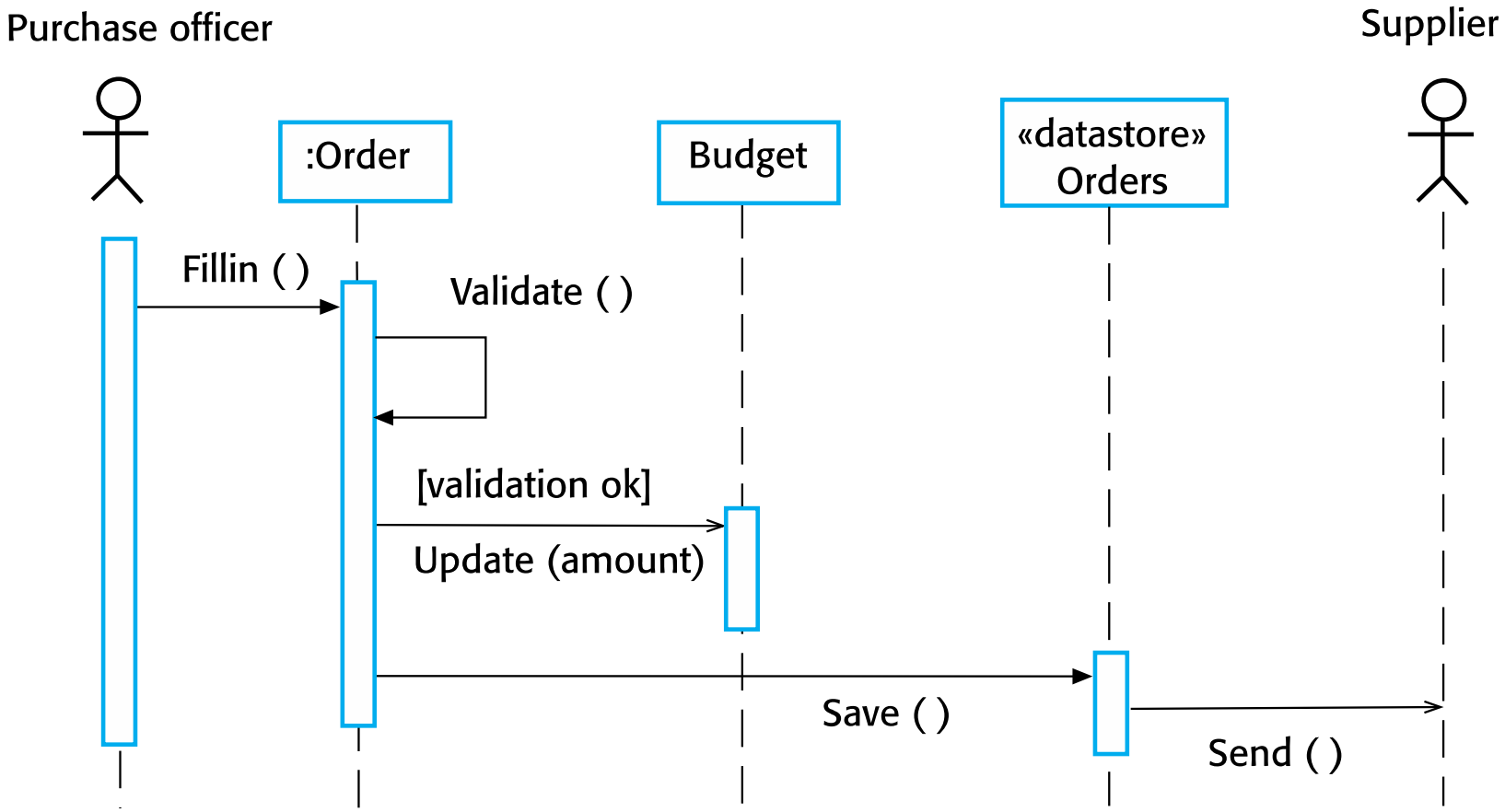
Data-driven Modeling

- 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.

An activity model of the insulin pump's operation



Order Processing



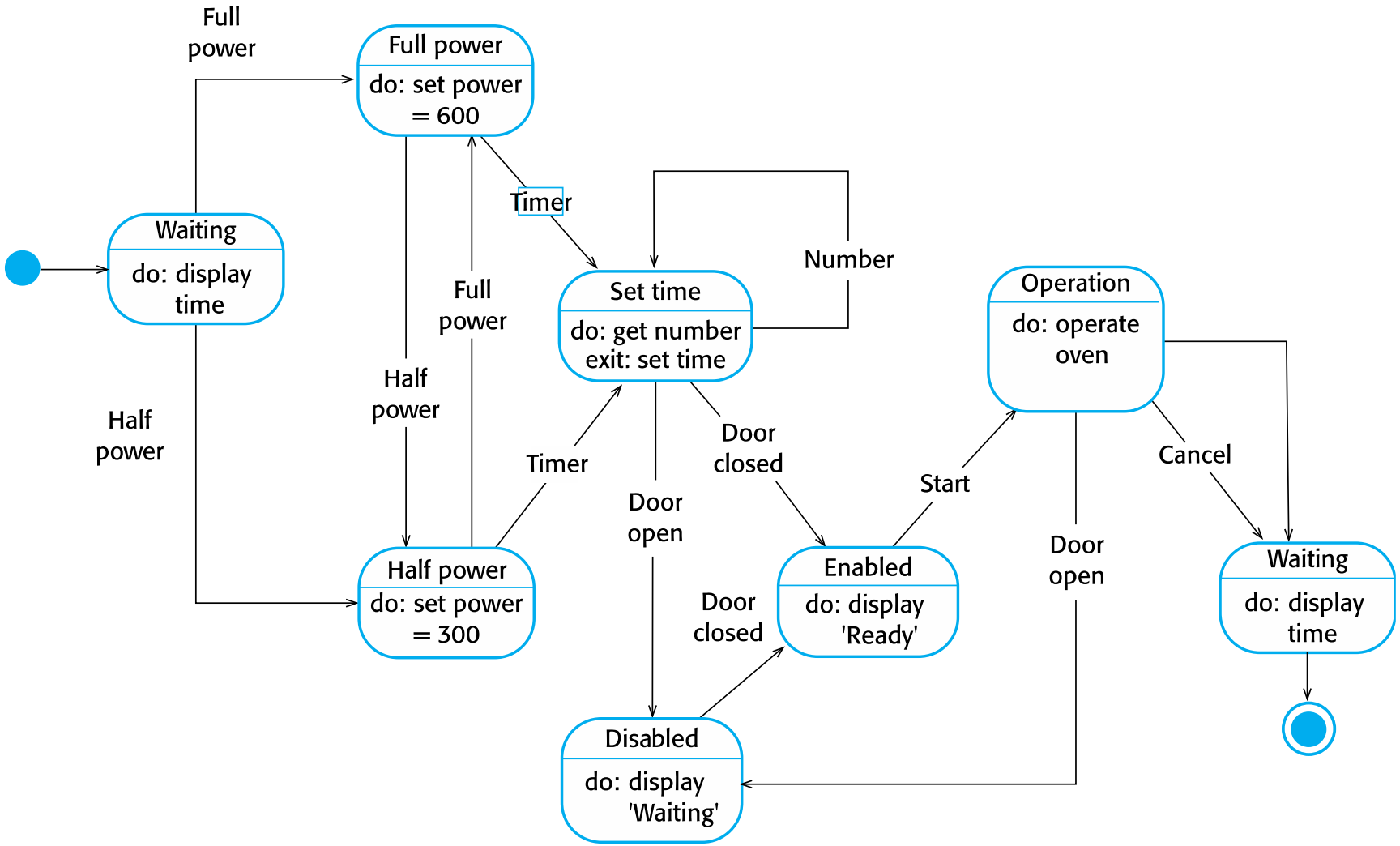
Event-driven Modeling

- 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.

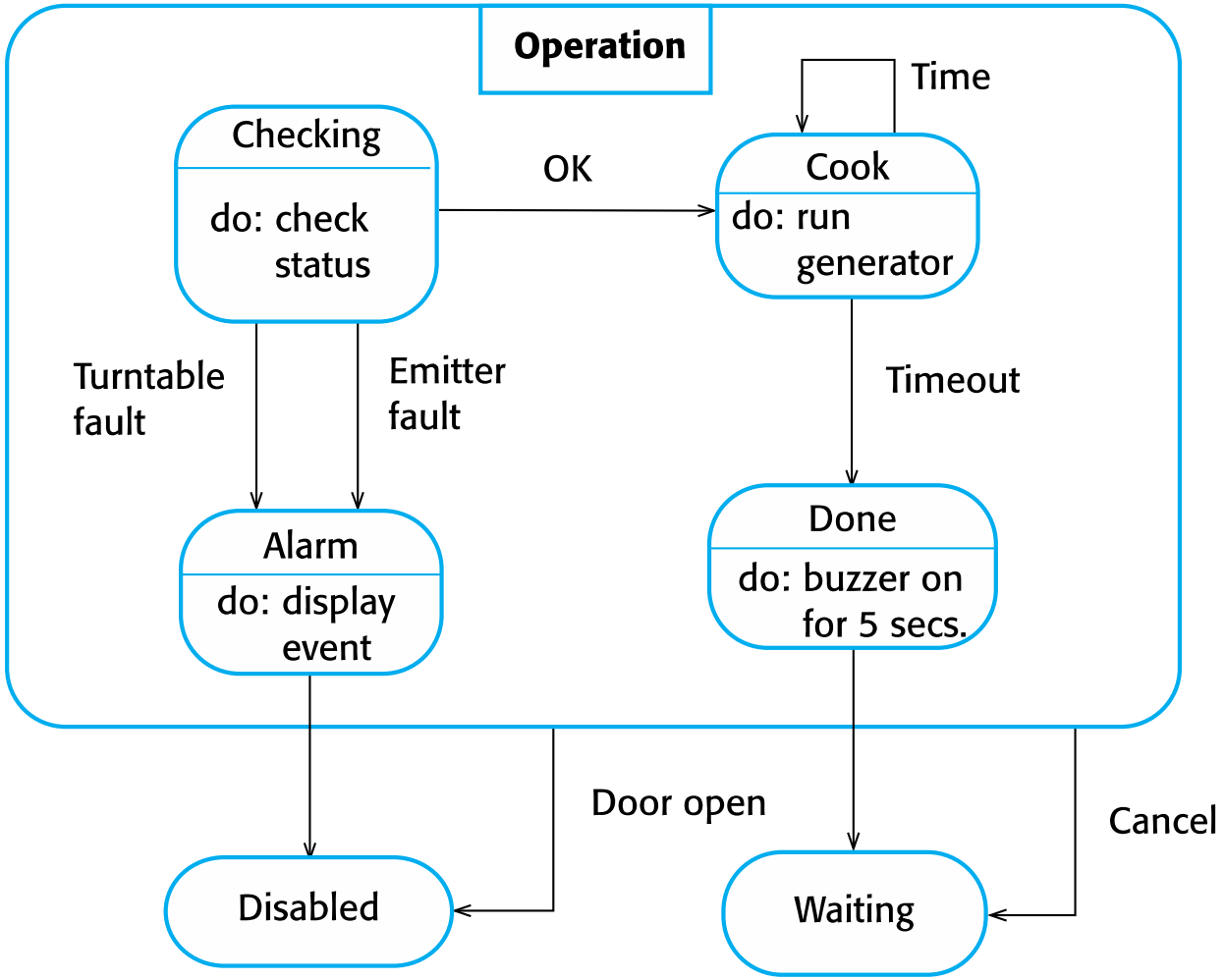
State Machine Models

- 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.
- Statecharts are an integral part of the UML and are used to represent state machine models.

State diagram of a microwave oven



Microwave oven operation



States and stimuli for the microwave oven (a)

State	Description
Waiting	The oven is waiting for input. The display shows the current time.
Half power	The oven power is set to 300 watts. The display shows ‘Half power’.
Full power	The oven power is set to 600 watts. The display shows ‘Full power’.
Set time	The cooking time is set to the user’s input value. The display shows the cooking time selected and is updated as the time is set.
Disabled	Oven operation is disabled for safety. Interior oven light is on. Display shows ‘Not ready’.
Enabled	Oven operation is enabled. Interior oven light is off. Display shows ‘Ready to cook’.
Operation	Oven in operation. Interior oven light is on. Display shows the timer countdown. On completion of cooking, the buzzer is sounded for five seconds. Oven light is on. Display shows ‘Cooking complete’ while buzzer is sounding.

States and stimuli for the microwave oven (b)

Stimulus	Description
Half power	The user has pressed the half-power button.
Full power	The user has pressed the full-power button.
Timer	The user has pressed one of the timer buttons.
Number	The user has pressed a numeric key.
Door open	The oven door switch is not closed.
Door closed	The oven door switch is closed.
Start	The user has pressed the Start button.
Cancel	The user has pressed the Cancel button.