CMPE 331 Software and Software Engineering

Dr. İlknur Dönmez

(Source: Pressman, R. *Software Engineering: A Practitioner's Approach*. McGraw-Hill, 2005
& Sommerville, I. (2004). Software Engineering. International computer science series. *ed: Addison Wesley*)

Topics covered

- Context models
- Interaction models
- Structural models
- Behavioral models
- Model-driven engineering

System modeling

- System modeling is the process of developing abstract models of a system,
- Each model presenting a different view or perspective of that system.
- System modeling is now representing a system using some kind of graphical notation, which is now almost always based on the Unified Modeling Language (UML).
- System modelling helps the analyst to understand the functionality of the system
- 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
  - They help to understand its strengths and weaknesses.
  - These then lead to requirements for the new system.

Existing and planned system models

- Models help to explain the proposed requirements to other system stakeholders.
- Engineers use these models to discuss design proposals
- Engineers use to document the system for implementation. In a model-driven engineering process, system implementation is generated from the system model.

System perspectives

- An external perspective, where you model the context or environment of the system. (What is the environment for the system)
- An interaction perspective, where you model the
  - interactions between a system and its environment,
  - interactions between the components of a system.
- A structural perspective, where you model the
  - organization of a system or
  - the structure of the processed data of 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
  - interactions 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.

Use of graphical models

- If we use models for discussion about the system
  - Incomplete and incorrect models are OK as their role is to support discussion.
- If we use models for documenting an existing system
  - Models should be an accurate representation of the system but need not be complete.
- If we use models as detailed system description to generate a system implementation
  - Models have to be both correct and complete.

Context models

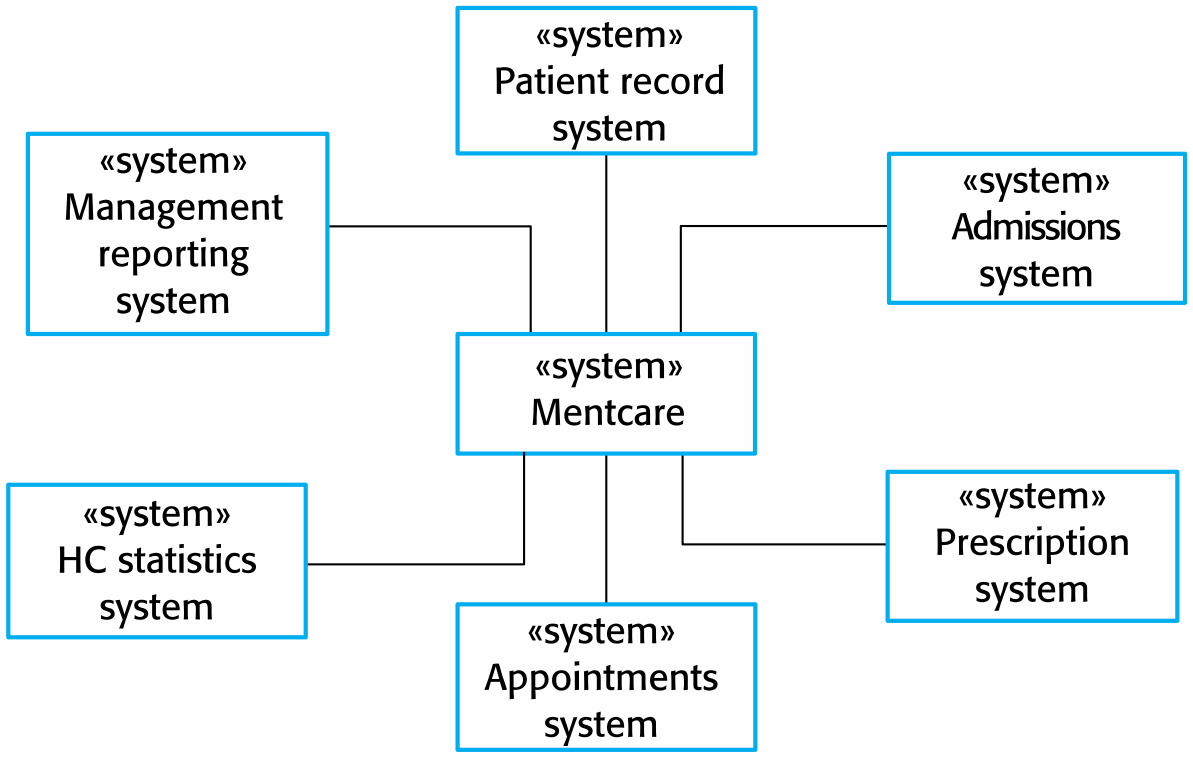
Context models

- 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 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 using the developed system or that are depend on the developed system.
- The position of the system boundary has a big effect on the system requirements.
- it increases / decreases the influence or workload of different parts.

The context of the Mentcare system


«system» Patient record system
«system» Management reporting system
«system» Admissions system
«system» Mentcare
«system» HC statistics system
«system» Prescription system
«system» Appointments system

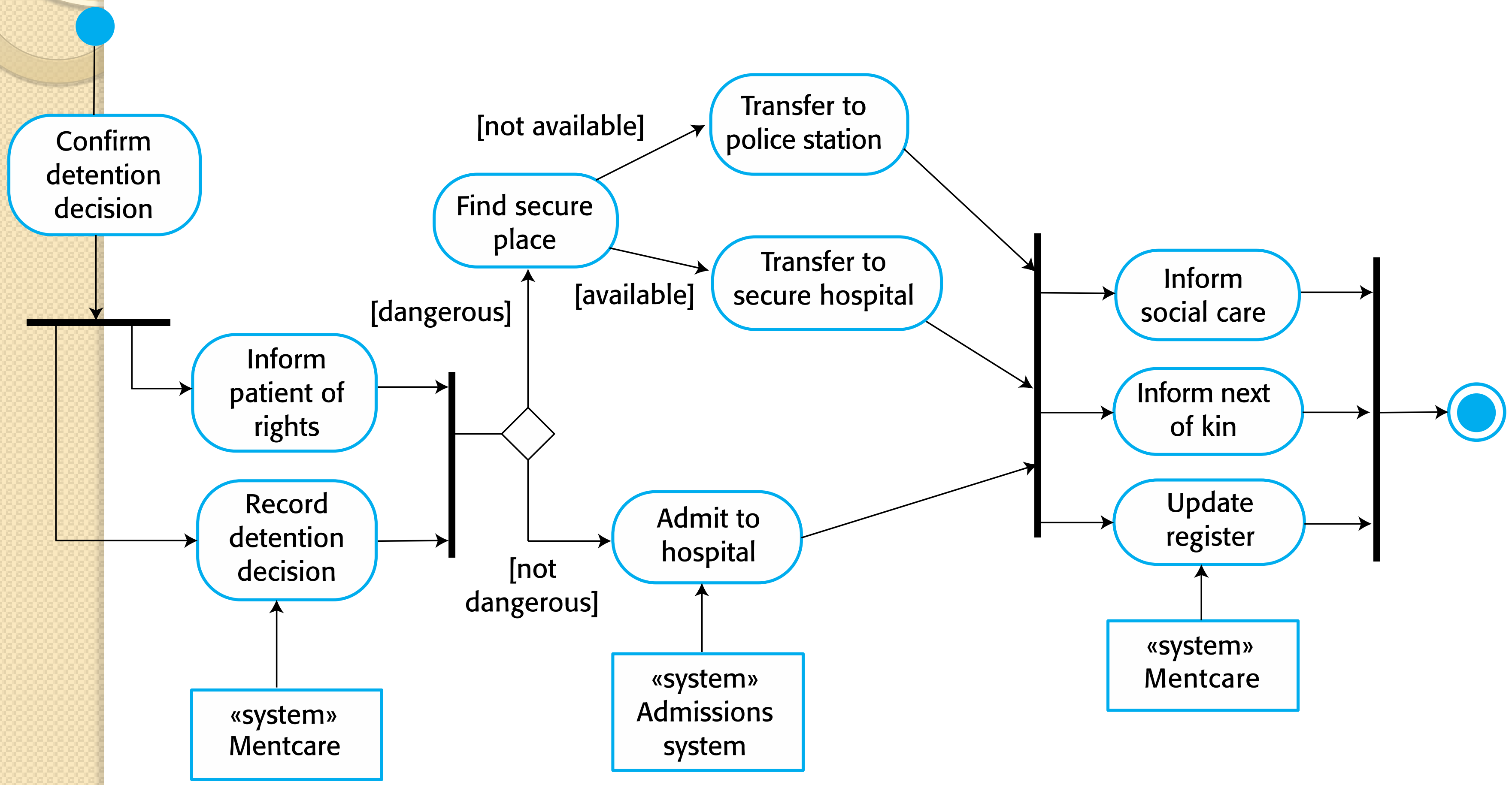
Example

- e-identity card system

Process perspective

- Context models simply show the other systems in the environment.
- But process models are related 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

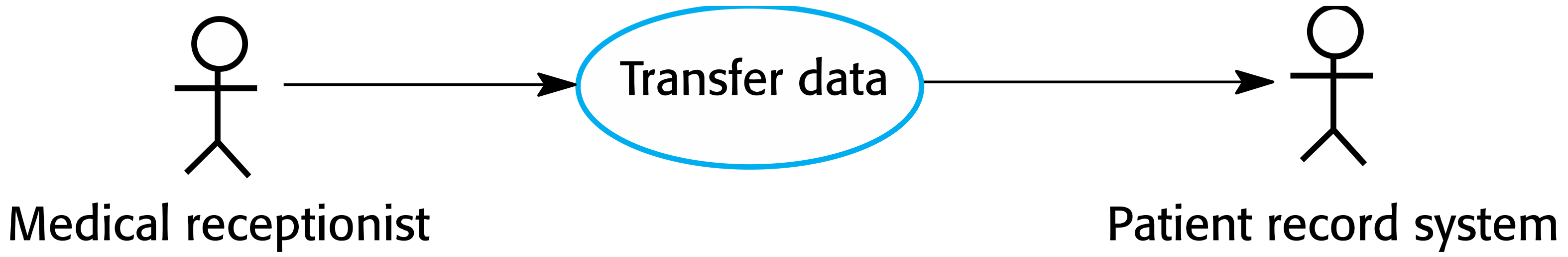
- Modeling user interaction is important. It helps to identify user requirements.
- Modeling system-to-system interaction highlights the communication problems that may arise.
- Modeling component interaction helps us understand if a proposed system structure meets the required system performance and 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 involved in 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.
- Representing it with diagram provides an overview of the use case and more detailed textual forms can be used.

Transfer-data use case

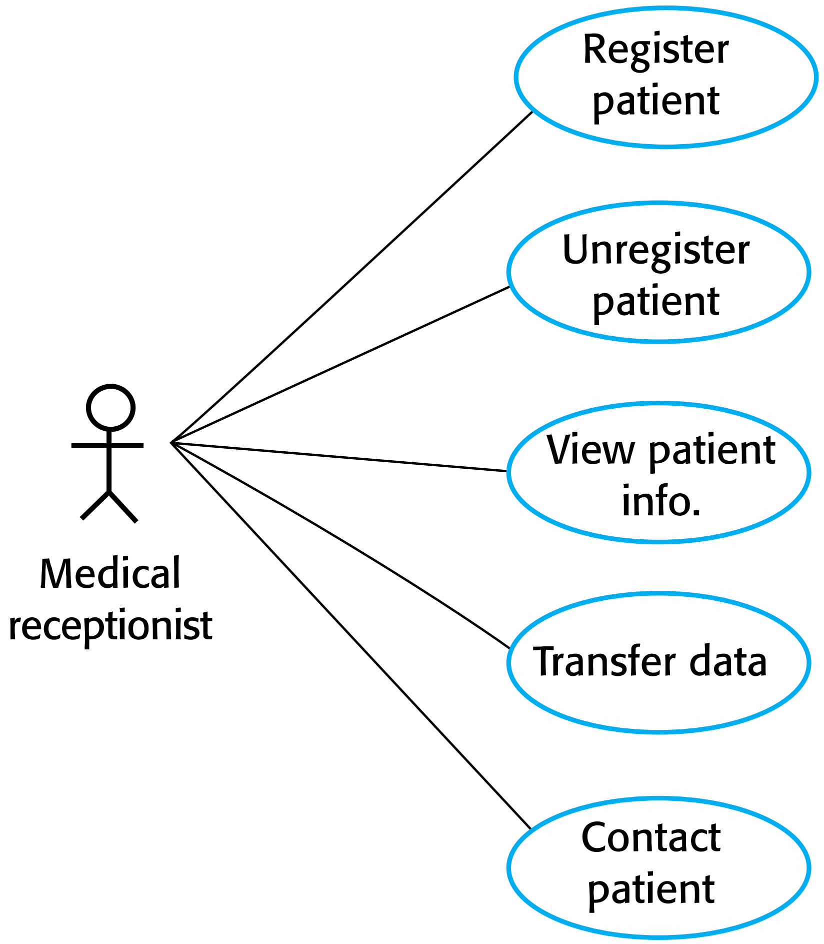
- A use case in the Mentcare system



Tabular description of the 'Transfer data' use-case

MHC-PMS: Transfer data	
Actors	Medical receptionist and authority using patient records system (PRS)
Description	A receptionist may transfer data from the Mentcase system to authority via a patient record database. The information transferred may 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 patient records system 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'



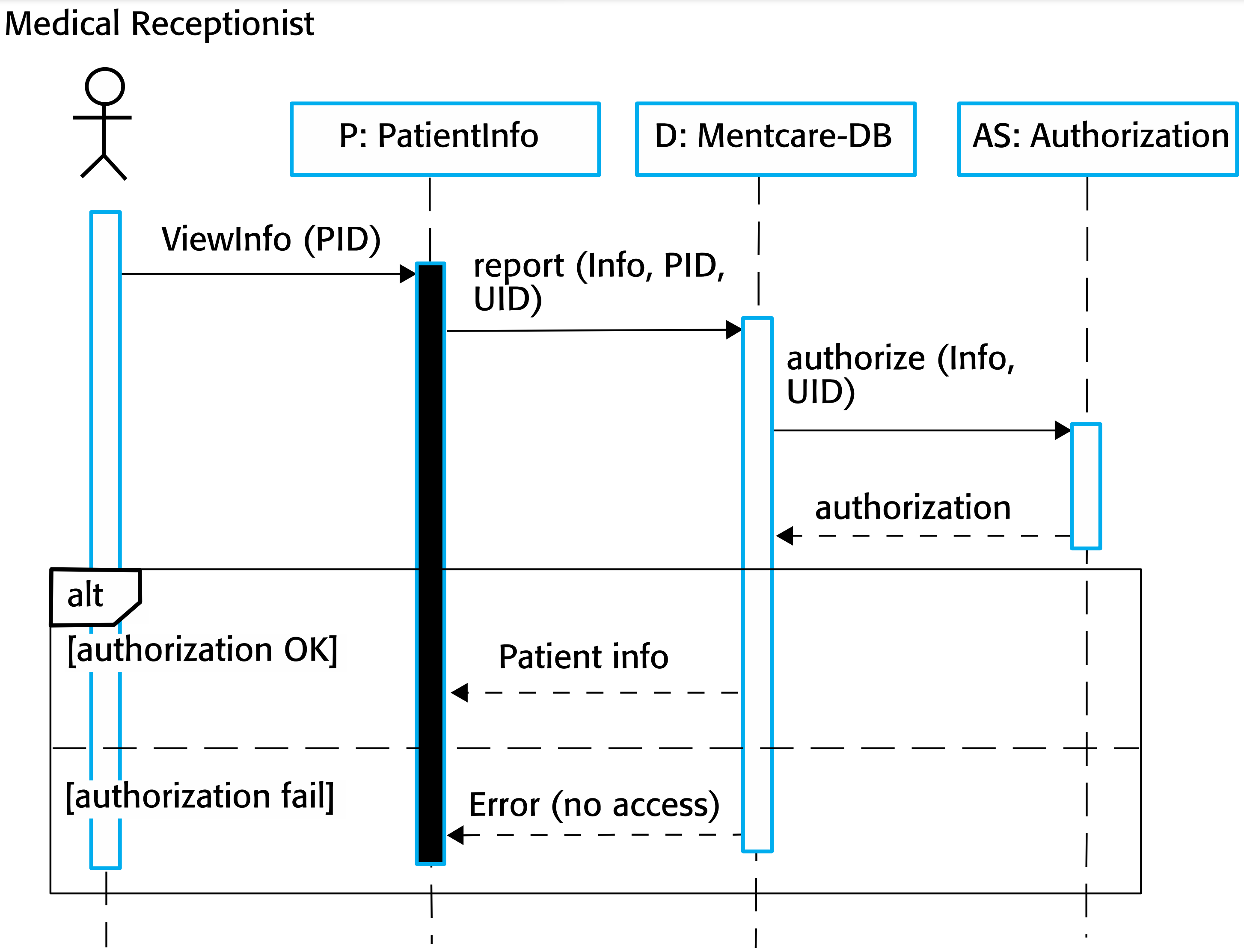
Example

- Write 5 use case for a television system

Sequence diagrams

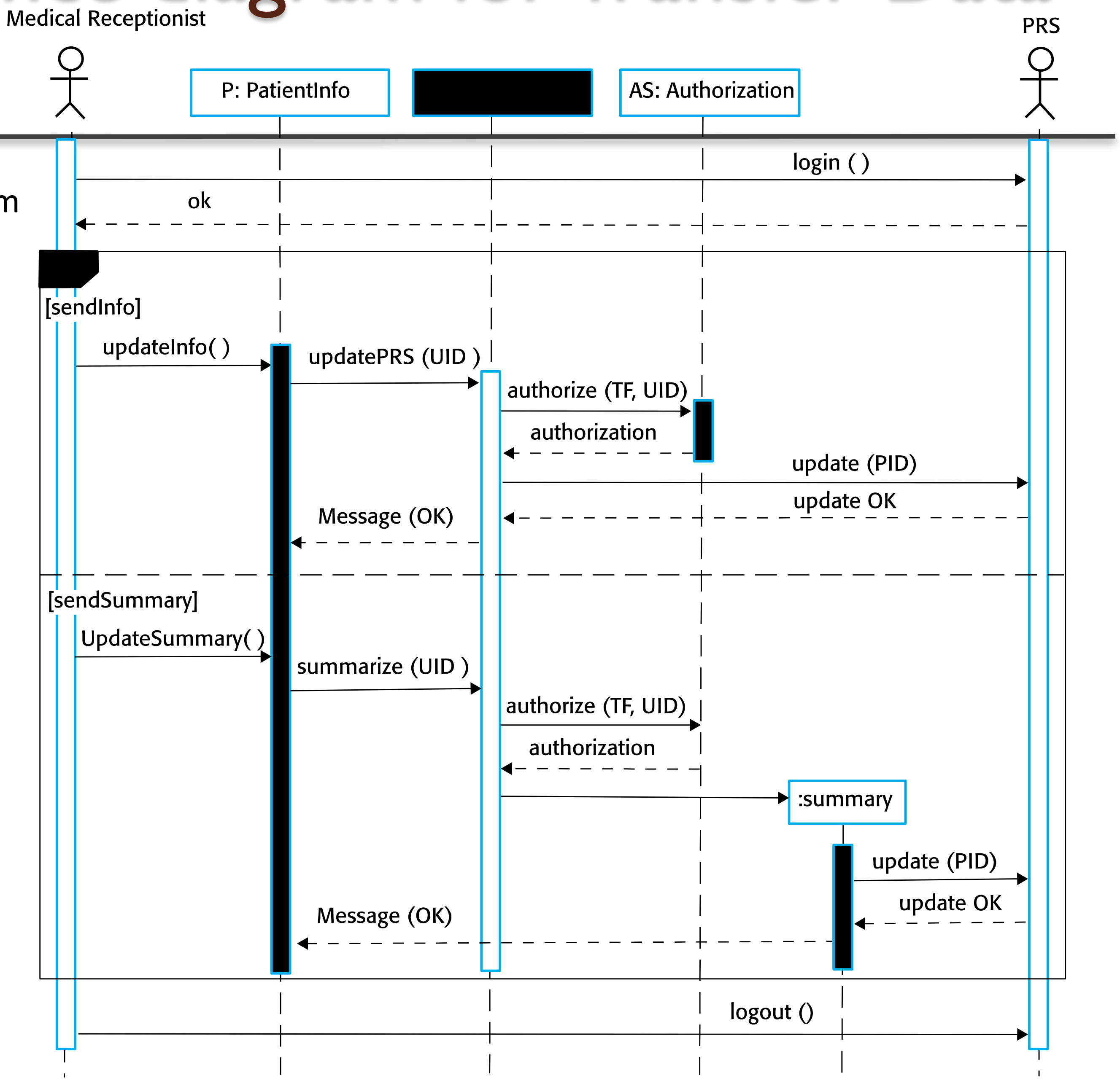
- Sequence diagrams are part of the UML and are used to model the interactions between the actors and the objects within a system.
- A sequence diagram shows the sequence of interactions that take place during a particular use case or use case instance.
- The objects and actors involved are listed along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

Sequence diagram for View patient information



Sequence diagram for Transfer Data

PRS: patient records system
TF: Transfer Flow Data
UID: User ID
PID: Patient ID



Structural models

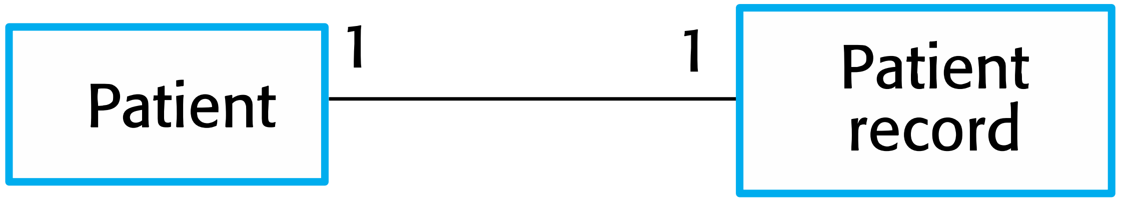
Structural models

- Structural models of software display the organization of a system with its components and their relationships.
- Structural models may be static models, which show the structure of the system design,
- Structural models may be dynamic models, which show the organization of the system when it is executing.
- Structural models of a system is used for discussing and designing the system architecture.

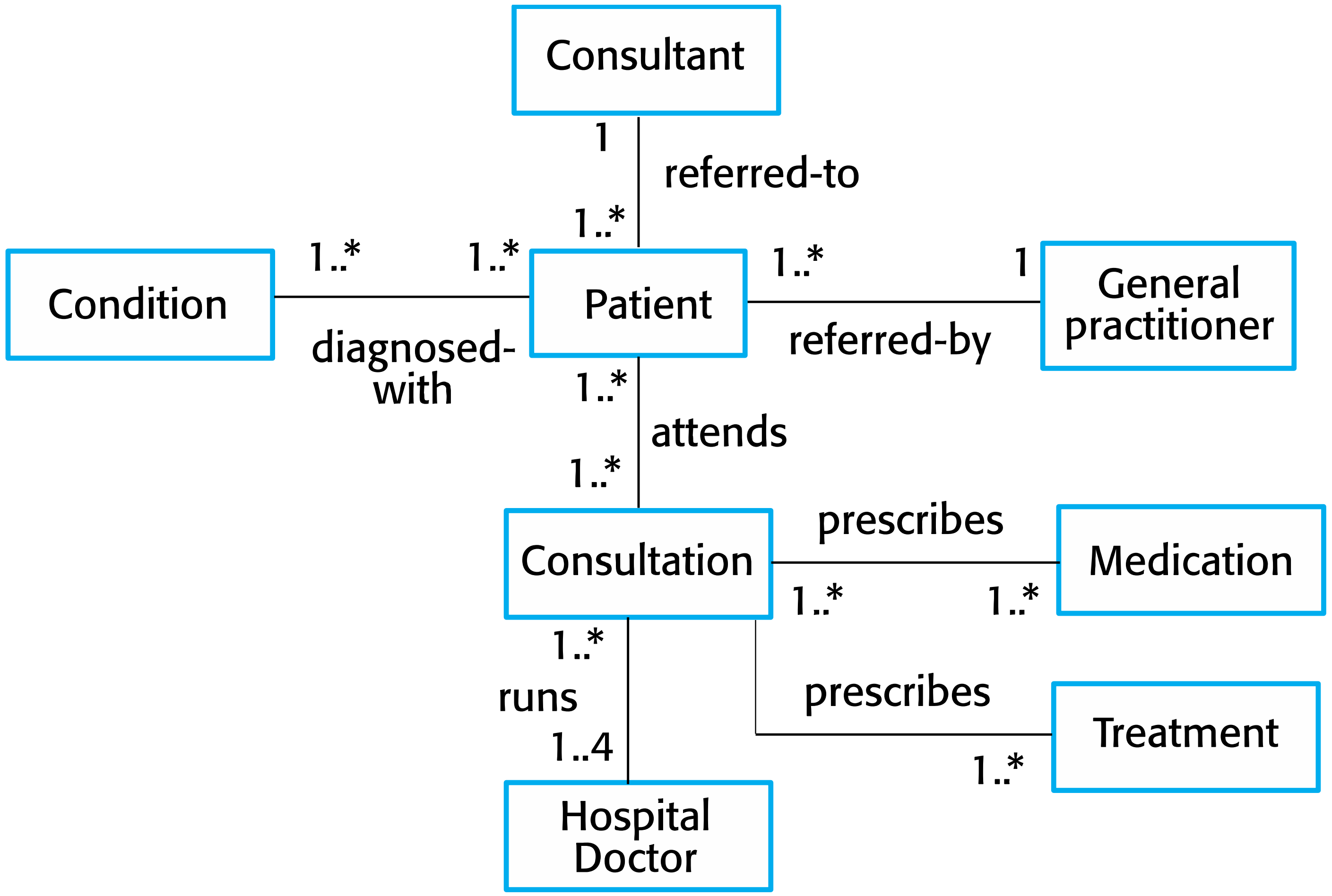
Class diagrams

- Class diagrams are used when developing an object-oriented systemç
- They show the classes in a system and the associations between these classes.
- An object class is 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, objects represent something in the real world, such as a patient, a prescription, doctor, etc.

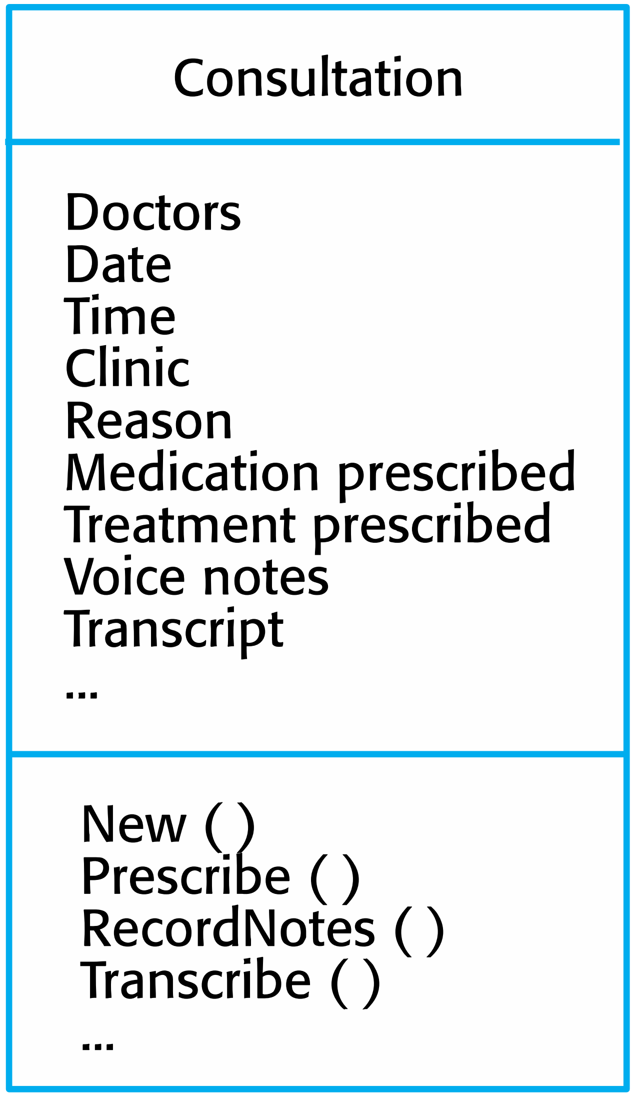
UML classes and association



Classes and associations in the MHC-PMS



The Consultation class



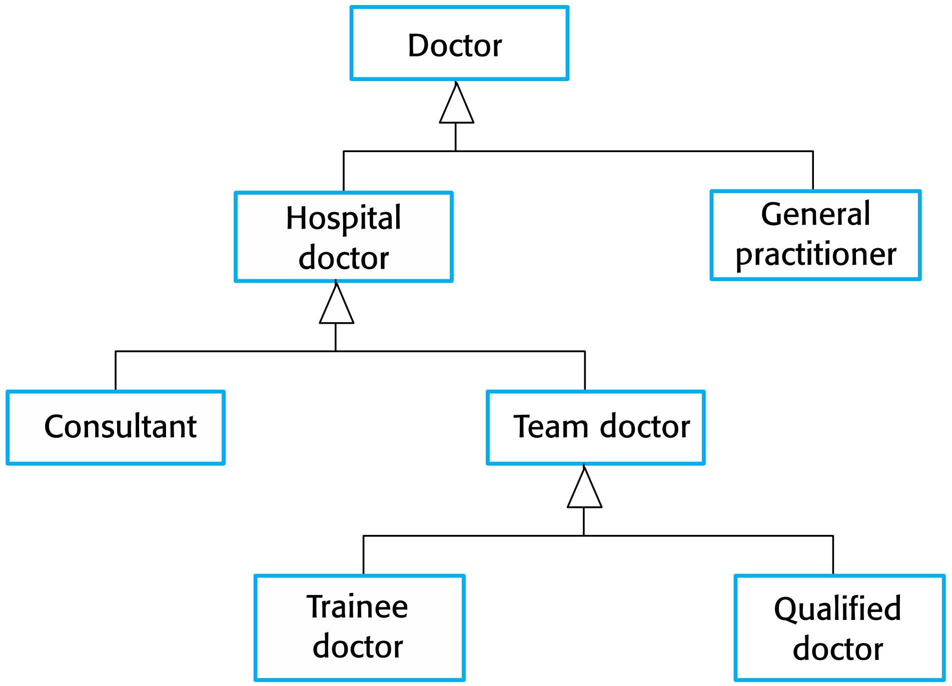
Class Diagrams & Generalization

- 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.
- The different members of these classes have some common characteristics e.g. Lions and crocodilles are carnivores.

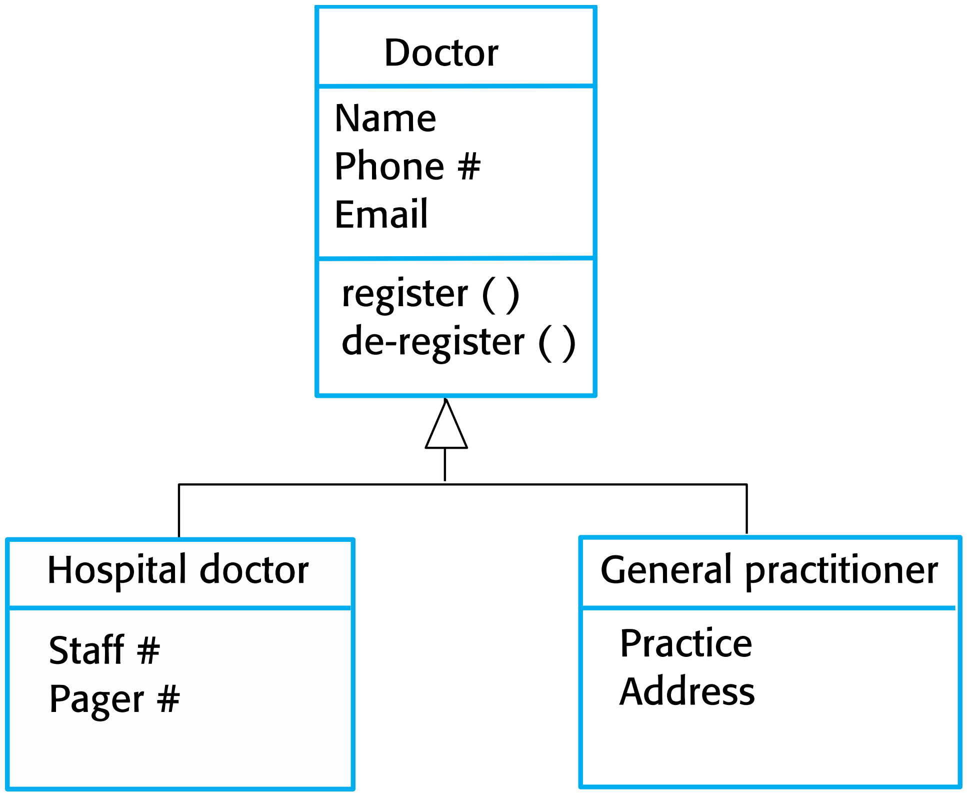
Class Diagrams & Generalization

- In modeling systems, if there is a way for generalization, do it. 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.
- 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. And they also have more specific attributes and operations.

A generalization hierarchy



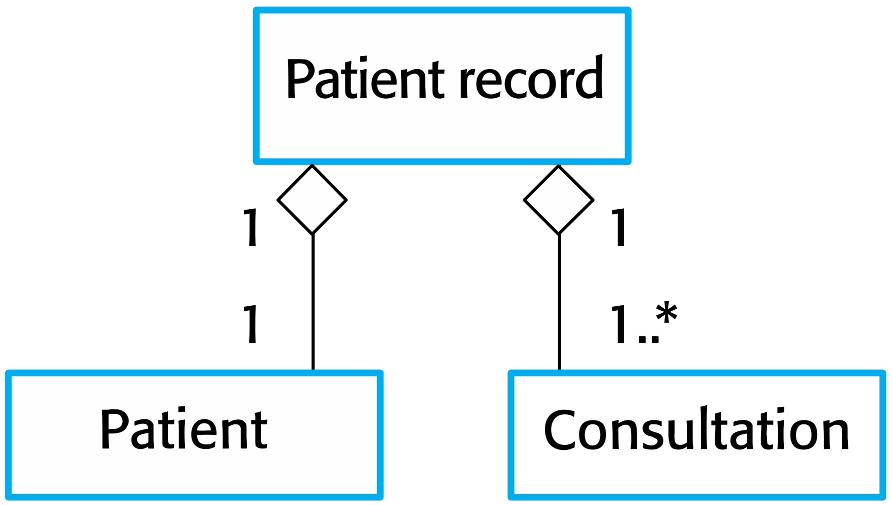
A generalization hierarchy with added detail



Object class aggregation models

- An aggregation model shows how classes that are collections are composed of other classes.
- Aggregation models show the relationship like semantic data models.

The aggregation association



Behavioral models

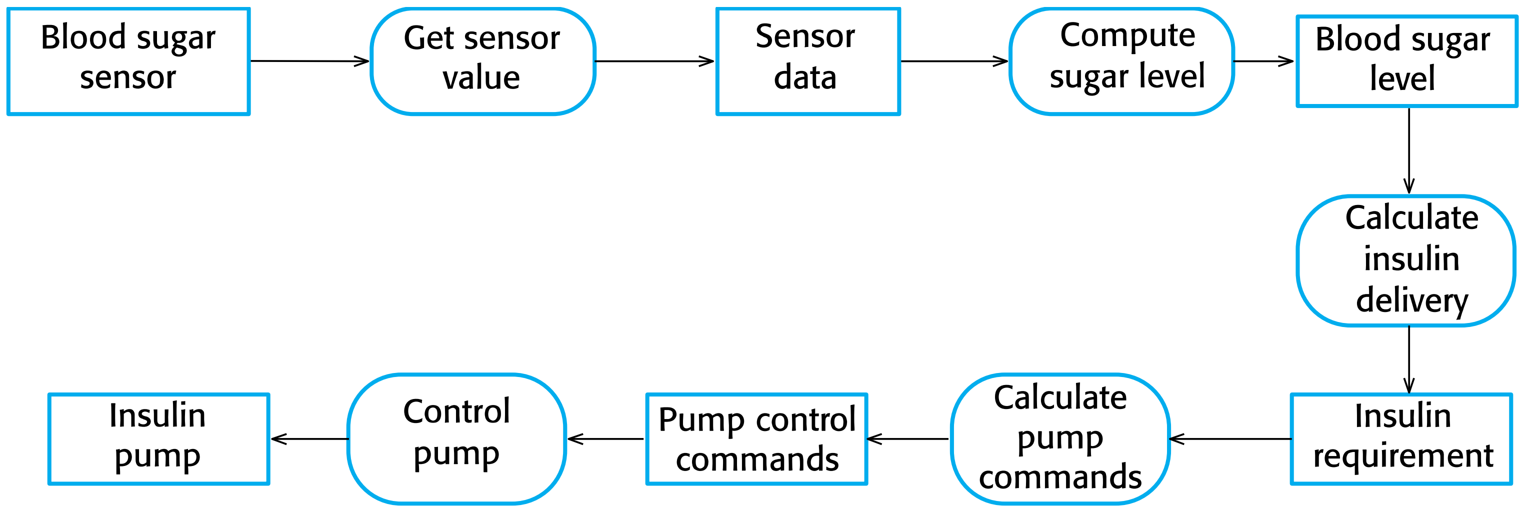
Behavioral models

- Behavioral models are models of the dynamic behavior of a system while it is executing.
- They show what happens or what is supposed to happen when a system responds to a stimulus from its environment.
- These stimuli are two types:
  - Data Some data arrives that has to be processed by the system.
  - Events Some event happens that triggers system processing.

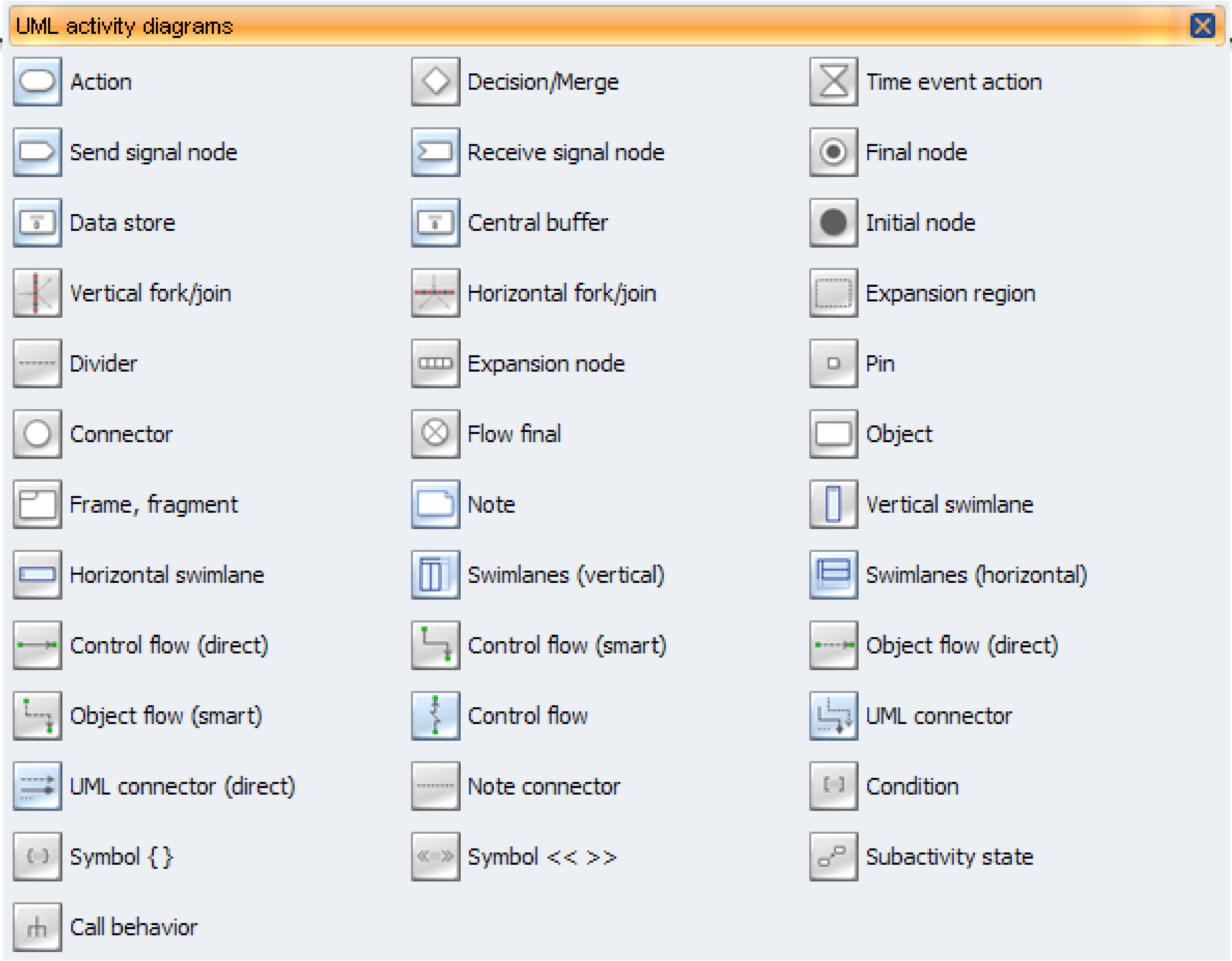
Data-driven modeling

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

An activity model of the insulin pump’s operation



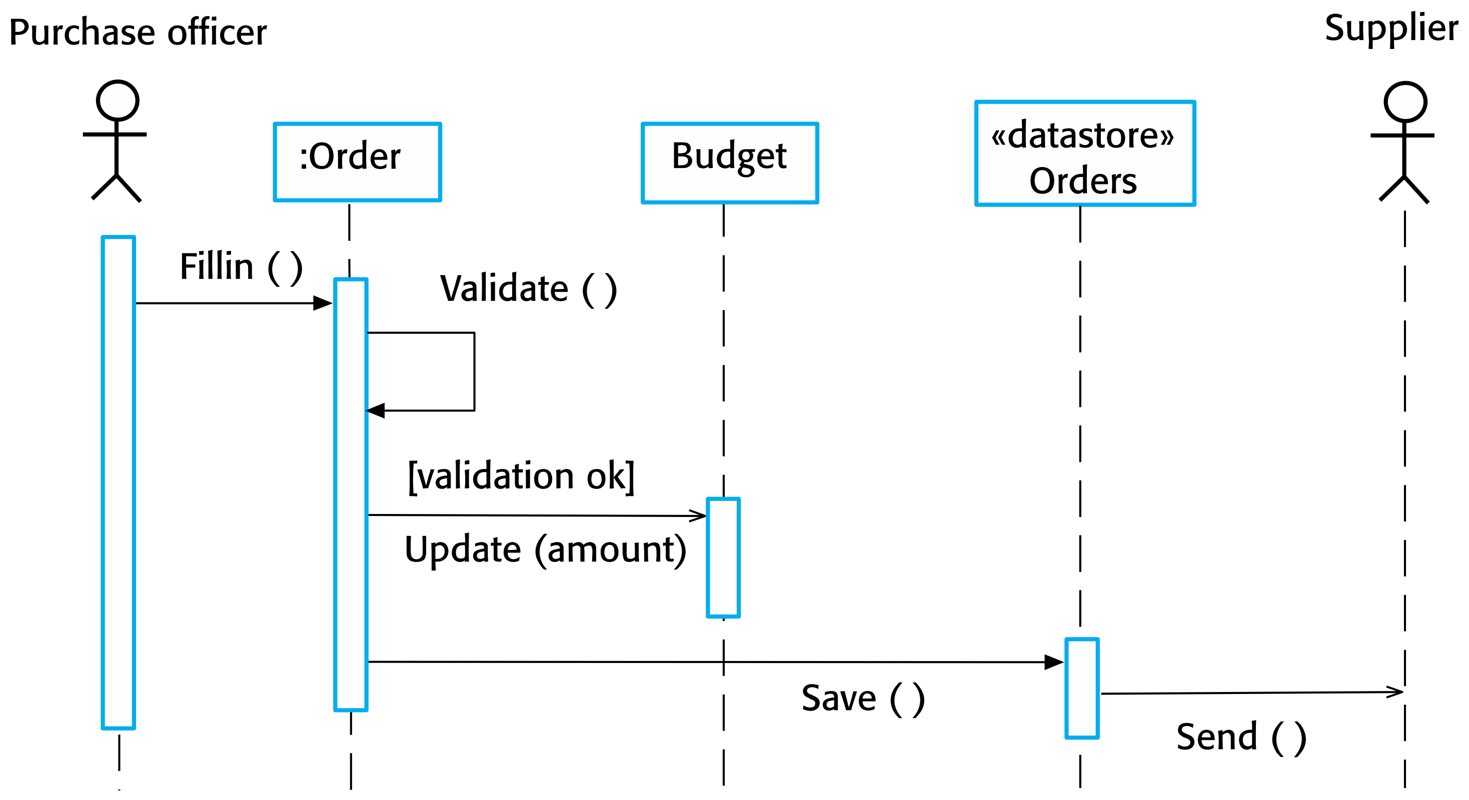
Activity Diagram Design Elements



UML activity diagrams

Action	Decision/Merge	Time event action
Send signal node	Receive signal node	Final node
Data store	Central buffer	Initial node
Vertical fork/join	Horizontal fork/join	Expansion region
Divider	Expansion node	Pin
Connector	Flow final	Object
Frame, fragment	Note	Vertical swimlane
Horizontal swimlane	Swimlanes (vertical)	Swimlanes (horizontal)
Control flow (direct)	Control flow (smart)	Object flow (direct)
Object flow (smart)	Control flow	UML connector
UML connector (direct)	Note connector	Condition
Symbol { }	Symbol << >>	Subactivity state
Call behavior		

Order processing Sequence Diagram



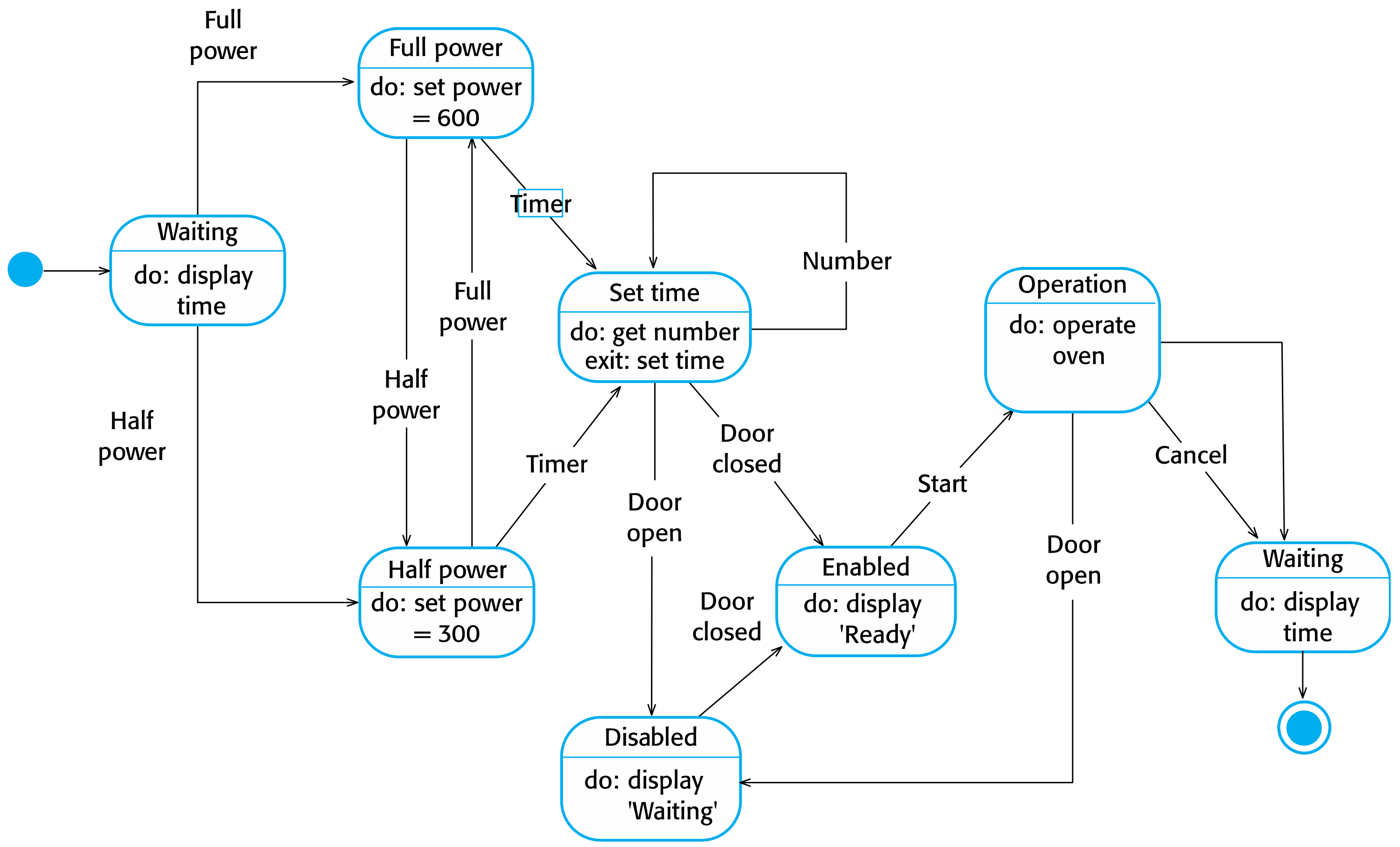
Event-driven modeling

- Real-time systems are often event-driven, with minimal data processing.
- 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.

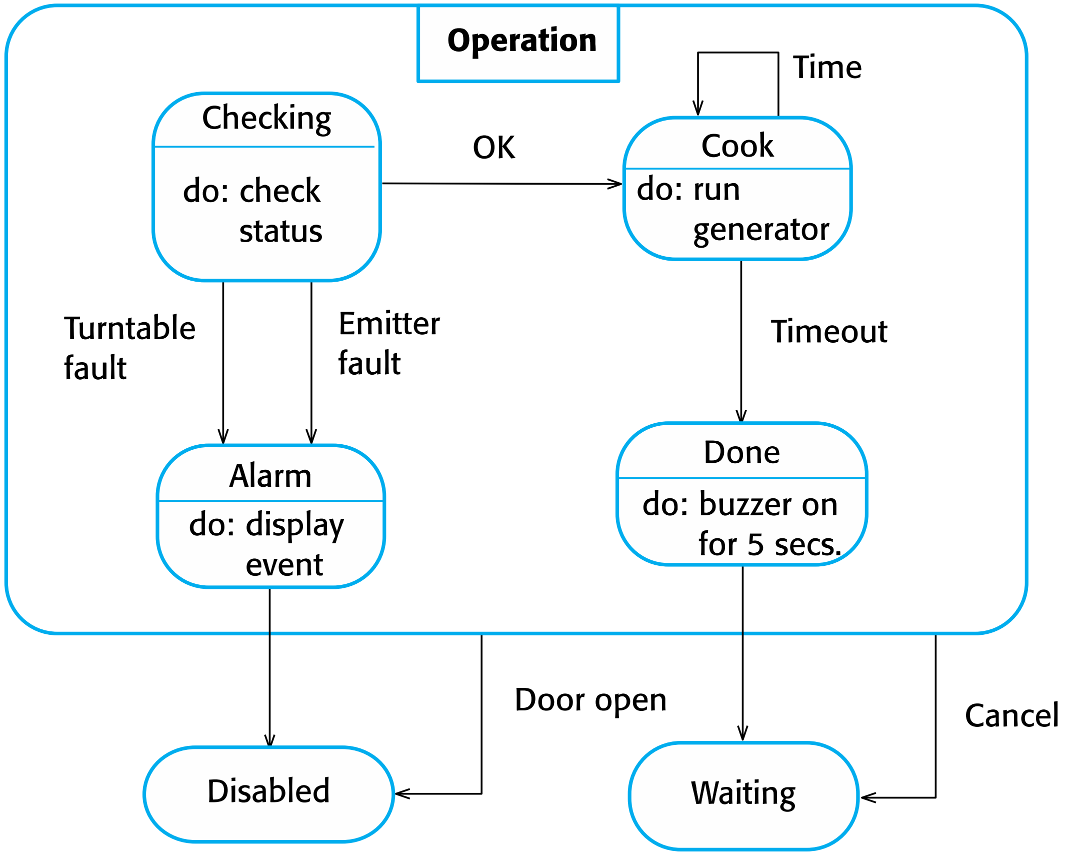
State machine models

- 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.
- To represent state machine models Statecharts of UML can be used.

State diagram of a microwave oven



Microwave oven operation



States and stimuli for the microwave oven (a)

State List		Stimulus List
Waiting		Half power
Half power		Full power
Full power		Timer
Set time		Number
Disabled		Door open
Enabled		Door closed
Operation		Start
		Cancel

Model-driven engineering

Model-driven engineering

- Model-driven engineering (MDE) is an approach to software development.
- Here the principal outputs of the development process are models rather than programs.
- The programs that execute on a hardware/software platform are then generated automatically from the models.
- Proponents of MDE argue that this raises the level of abstraction in software engineering.
- Engineers no longer have to be concerned with programming language details or execution platforms.

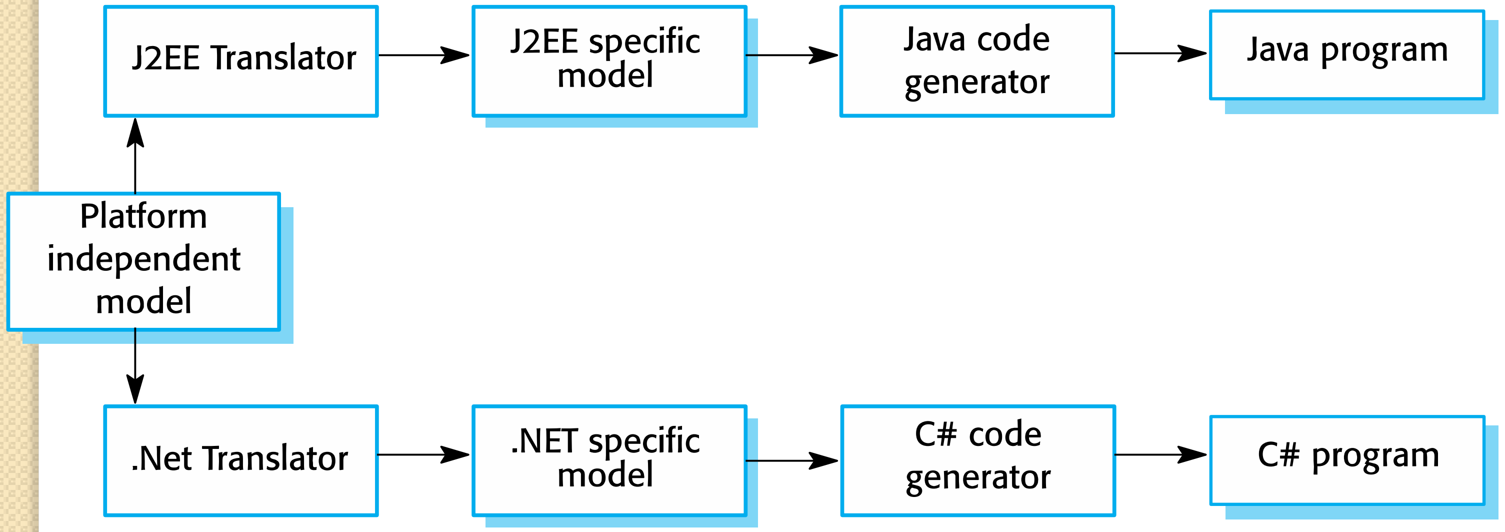
Usage of model-driven engineering

- Model-driven engineering is still at an early stage of development,
- It is unclear if it will have a significant effect on software engineering practice.
- Pros
  - Allows systems to be at higher levels of abstraction
  - It is cheaper to adapt systems to new platforms.
- Cons
  - Models for abstraction may not be right for implementation.
  - Developing translators for new platforms may cost more than expected.

Model driven architecture

- MDA is a model-focused approach to software design and implementation
- It uses a subset of UML models to describe a system.
- Models at different levels of abstraction are created.
- It is a high-level, platform independent model.
- It is possible to generate a working program without manual addition using this model.

Multiple platform-specific models


J2EE Translator
J2EE specific model
Java code generator
Java program
Platform independent model
.Net Translator
.NET specific model
C# code generator
C# program