1. System Modeling

System_Modeling

- process of developing abstract models of system, each model a different view
- representing system using graphical notation (UML)
- help analyst understand functionalities of system
- models used to communicate with customers

Existing and planned system models

Models of existing system

used during requirements engineering; help clarify what system does and used as basis for discussing strengths/weaknesses -> lead to requirements of new system

Models of new system

used during requirements engineering; help explain proposed requirements to stakeholders; engineers use to discuss design proposals and document system for implementation

- in model-driven engineering process, it is possible to generate complete/partial system implementation from system model

System perspectives

1. External perspective
2. Interaction perspective
3. .Structural perspective
4. Behavioral perspective

UML diagram types

1. Activity diagrams
2. Use case diagrams
3. Sequence diagrams
4. Class diagrams
5. State diagrams

Use of graphical models

- means of facilitating discussion about existing/proposed system
- way of documenting existing system
- detailed system description used to generate a system implementation

---

2. Context Models

Context_Models

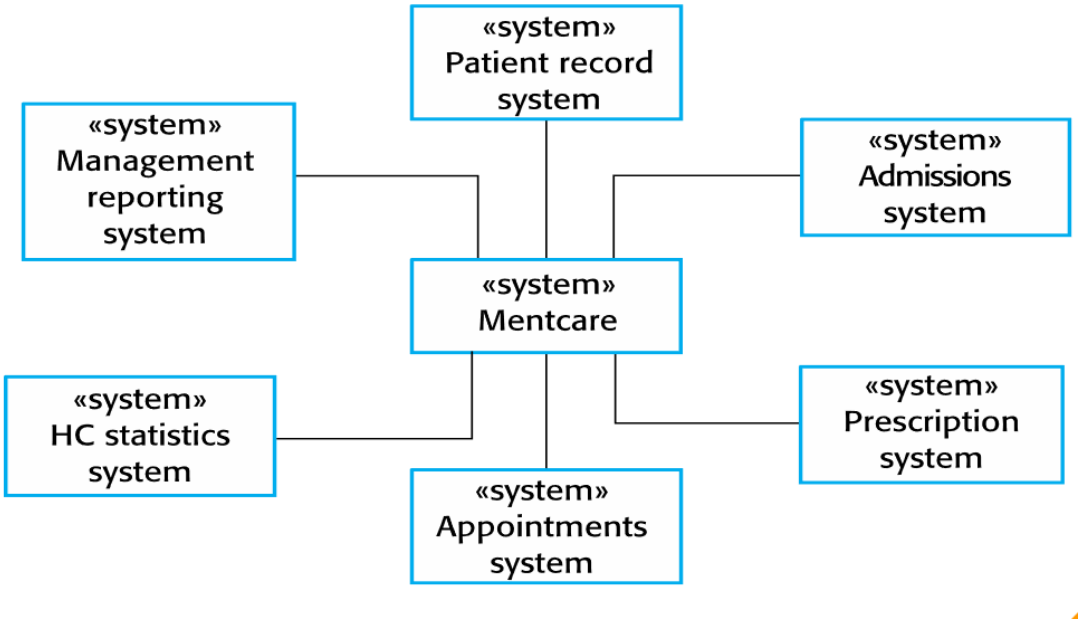
- illustrate operational context of system; show what lies outside *system boundaries*
- social/organisational concerns affect decision on where to position *system boundaries*
- architectural models **show system and relationship with other systems**

System boundaries

- established to define the inside and outside of system
    - show other systems used/dependent on system being developed
- position of system boundary has profound effect on system requirements
- defining system boundary is political judgment
    - pressures to develop system boundaries that increase/decrease influence/workload of different parts of organization

Context of Mentcare System

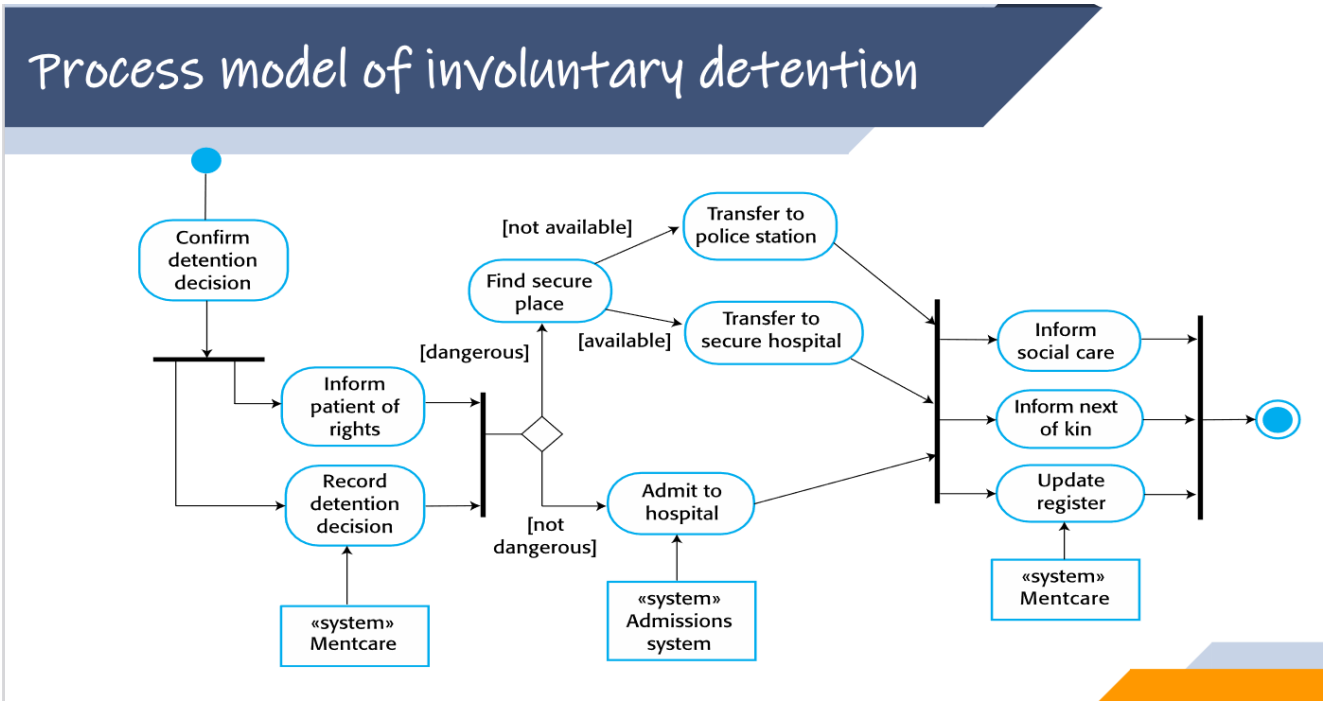
The context of the Mentcare system



Process perspective

- context models show other system in environment, now how system is used in environment
- process model reveal how system is used in broader business processes
- **UML activity diagrams** used to define business process models

Process model of involuntary detention



3. Interaction Models

Interaction_Models

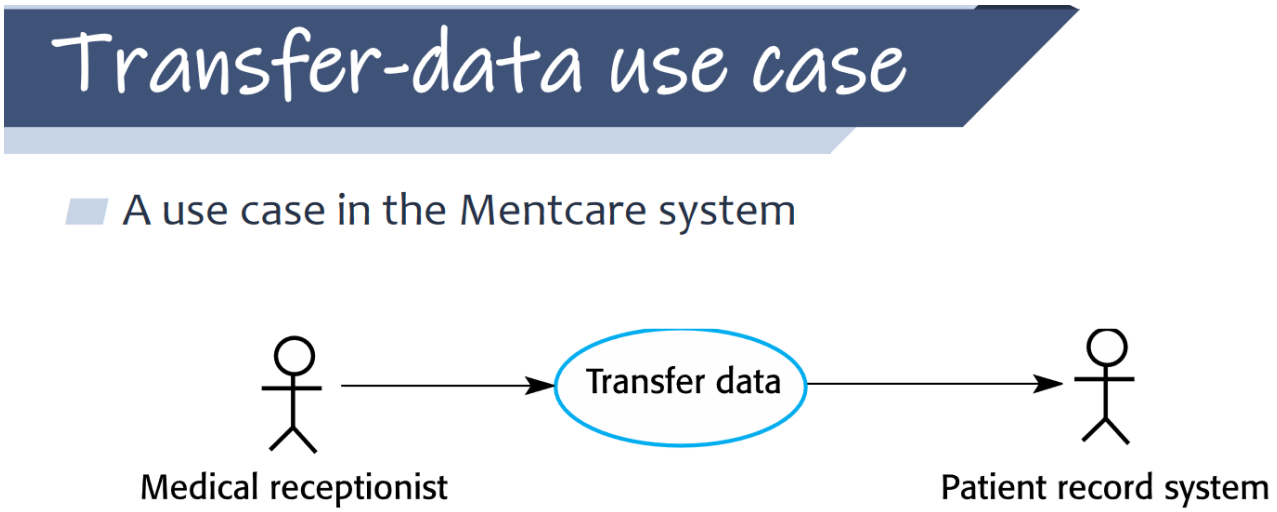
- modeling user interaction is important as it helps identity user requirements
- modeling system-to-system interaction highlights communication problems that arise
- modeling component interaction helps understand if proposed system structure is likely to deliver required system performance/dependability
- **Use case diagrams and Sequence diagrams** used for interaction modeling

Use case modeling

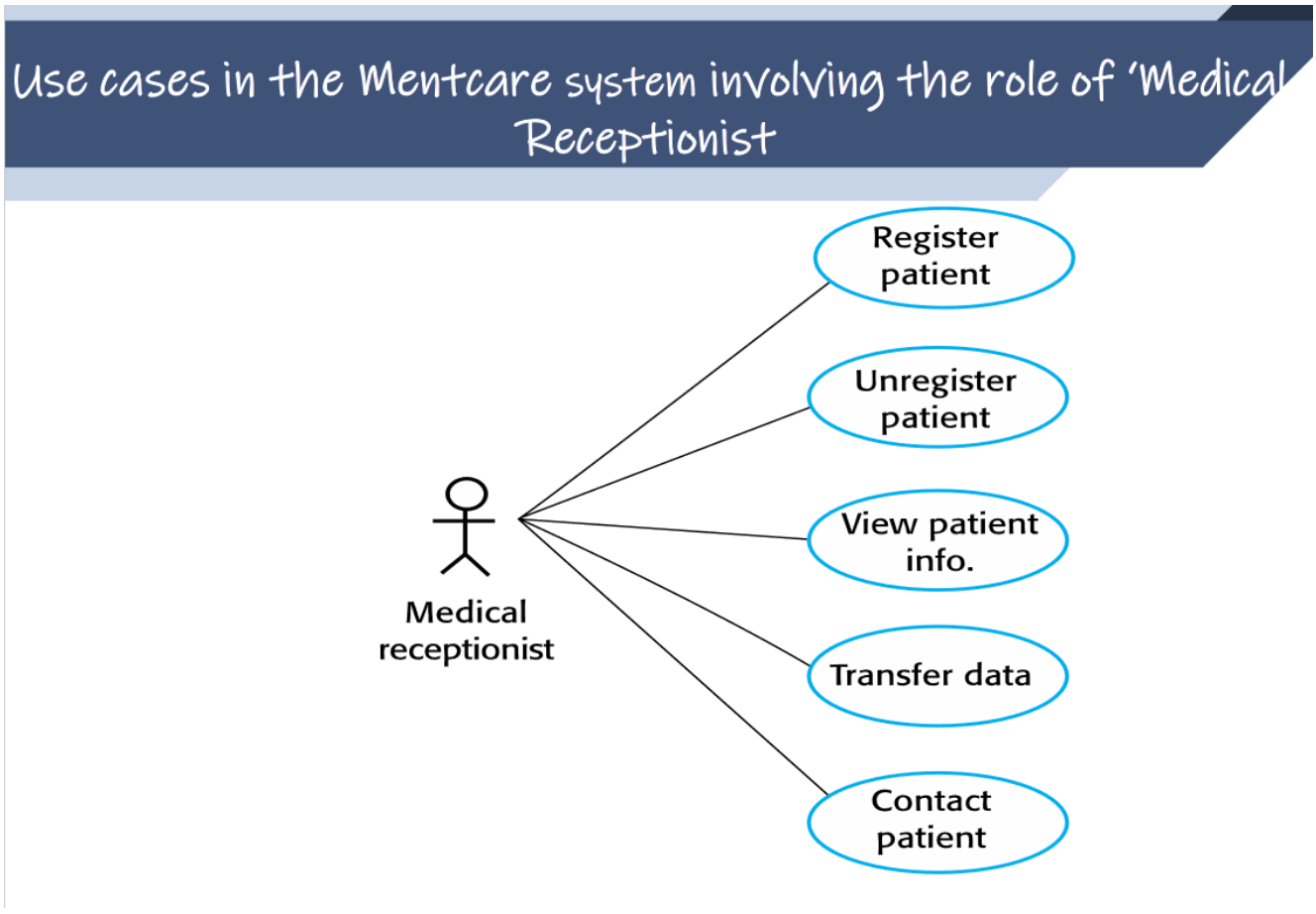
- use cases were developed originally to support requirements elicitation and incorporated into UML
- each use case represents a discrete task that involves external interaction with a system
- actors in a use case may be people or systems
- represented diagrammatically to provide an overview of use case in more detailed textual form

Use case/Sequence diagram examples

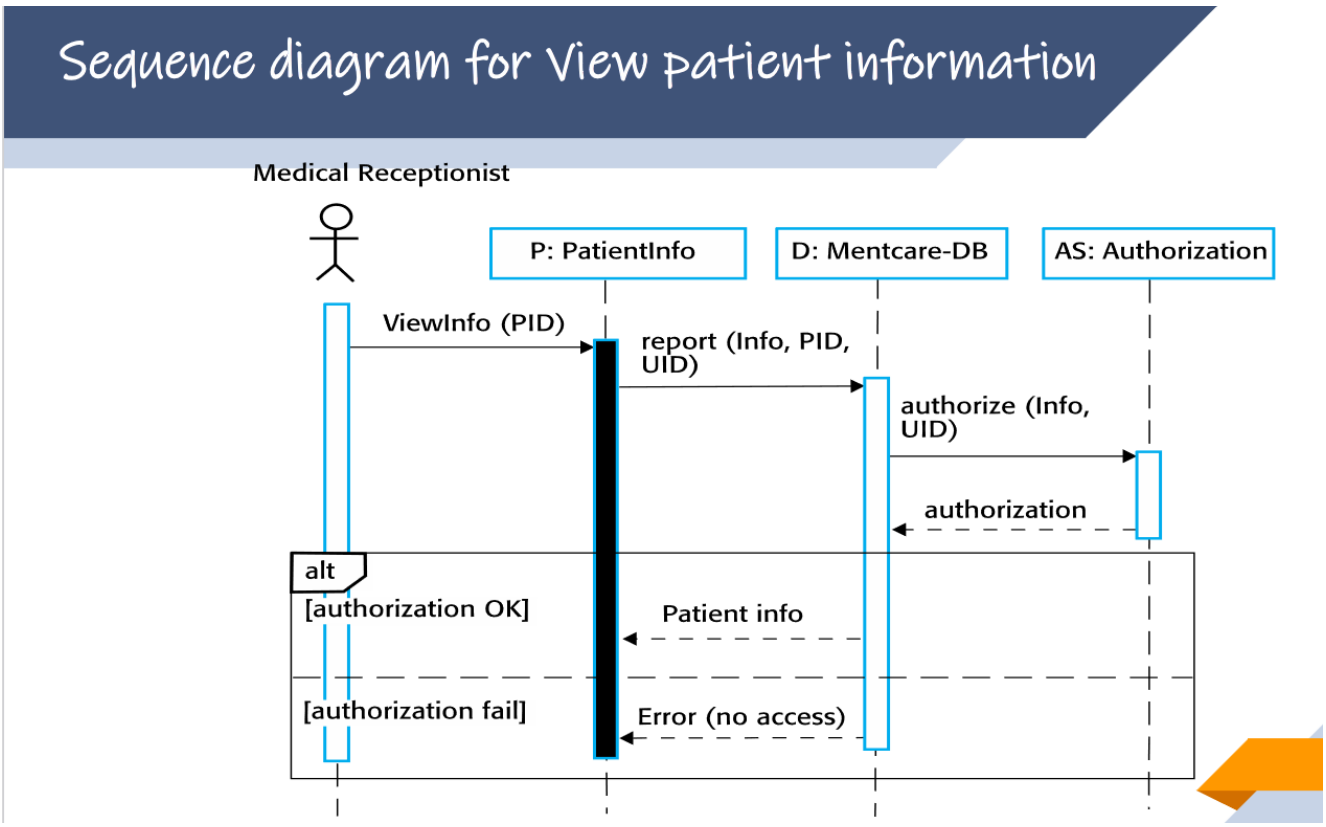
Transfer data use case



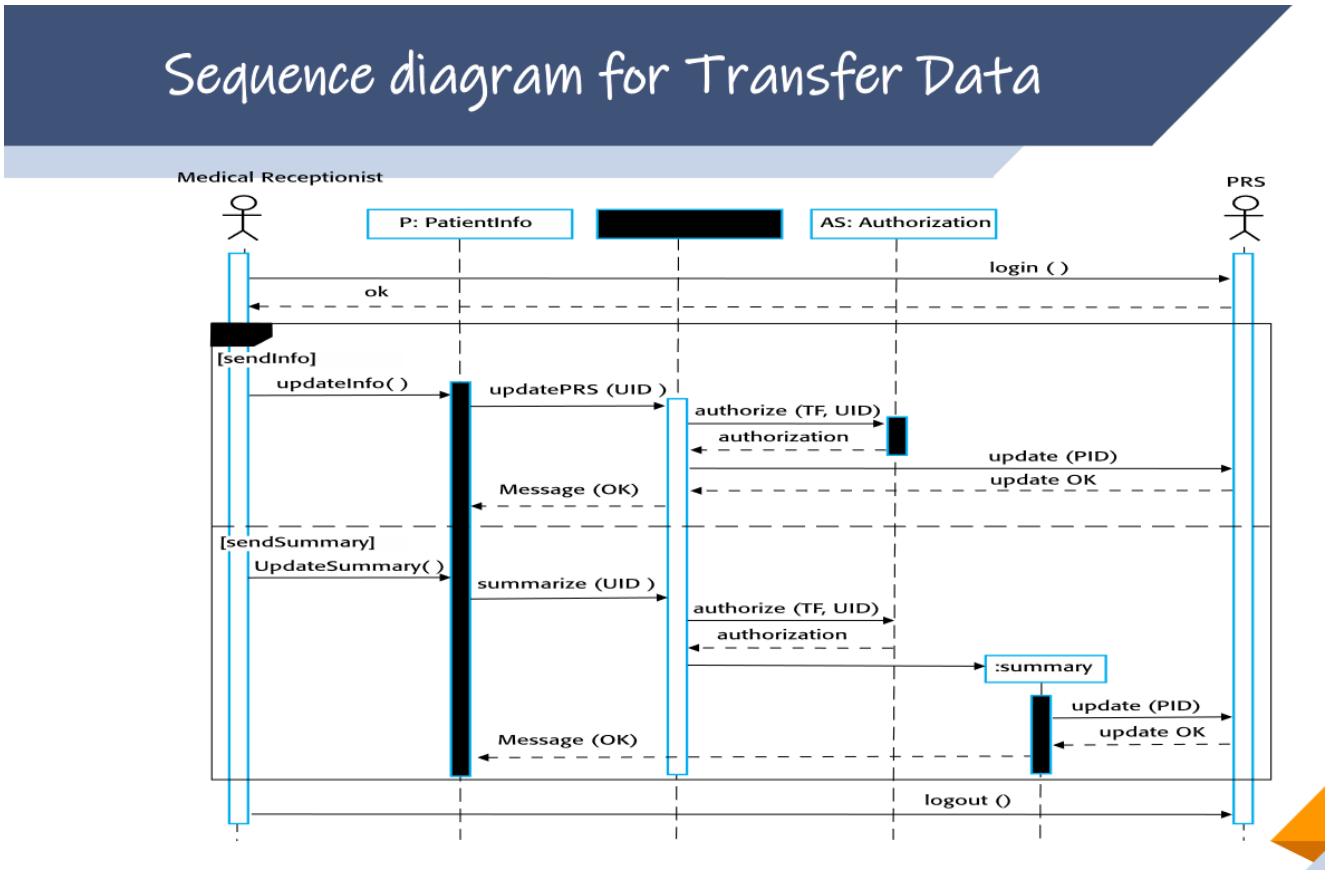
Use cases in Mentcare system involving Medical Receptionist



Sequence diagram for View patient information



Sequence diagram for Transfer Data



Sequence diagrams

- part of UML and used to model interactions between actors/objects within system
- shows sequence of interactions that take place during particular use case/instance
- objects/actors involved are listed along top of diagram, with dotted line drawn vertically
- interactions between objects are indicated by annotated arrows

---

4. Structural Models

Structural_Models

- display organization of system in terms of components/relationships
- may be **static models** that show structure of system design, or **dynamic models** that show organization of system when executing
- create structural models **when discussing and designing system architecture**

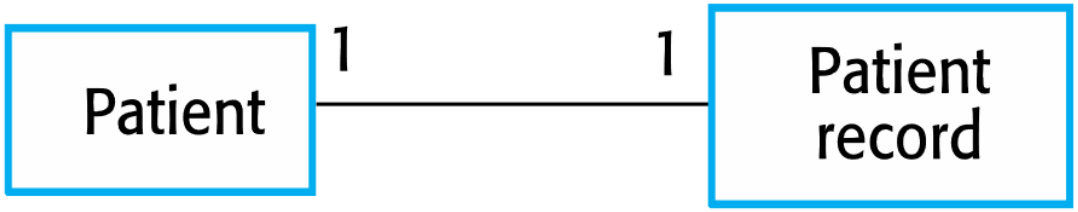
Class diagrams

- used when developiing object-oriented system model to show classes/associations in system
- **object class** is a general definition of one kind of system object
- **association** is a link/relationship between classes
- when developing models during early stages of software engineering process, objects represent something in real world (patient, prescription, doctor, etc.)

Class diagrams/Association examples

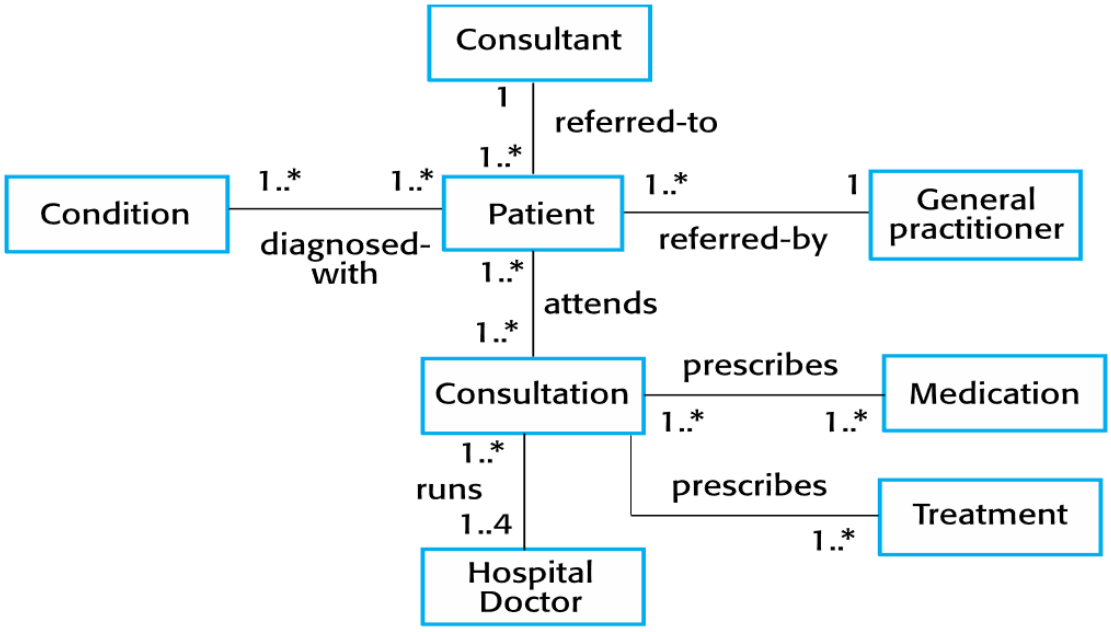
UML classes and association

UML classes and association



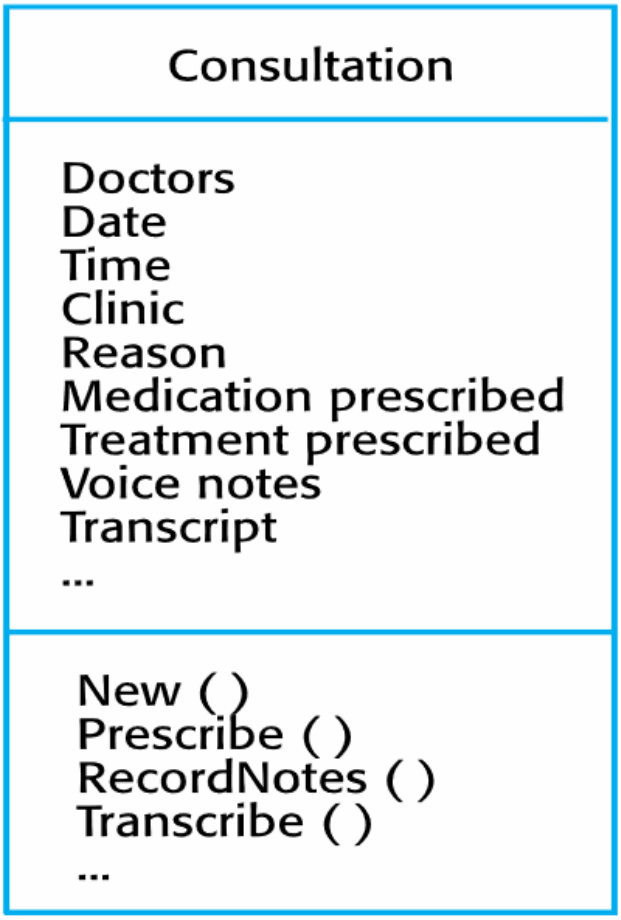
Classes and associations in MHC-PMS

Classes and associations in the MHC-PMS



Consultation class

The Consultation class



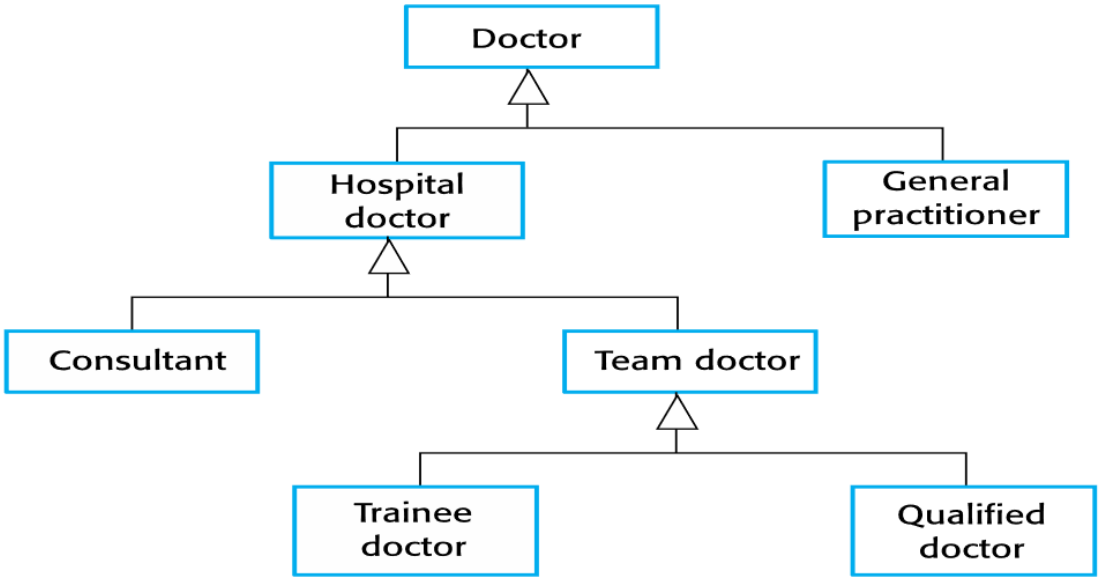
Generalization

- everyday thechnique used to manage complexity
- rather than learn details, place entities in more general classes (animals, cars, houses, etc.) and learn characteristics of classes
- allows to infer that different members of these classes have common characteristics (squirrels and rats are rodents)

Generalization hierarchy diagrams

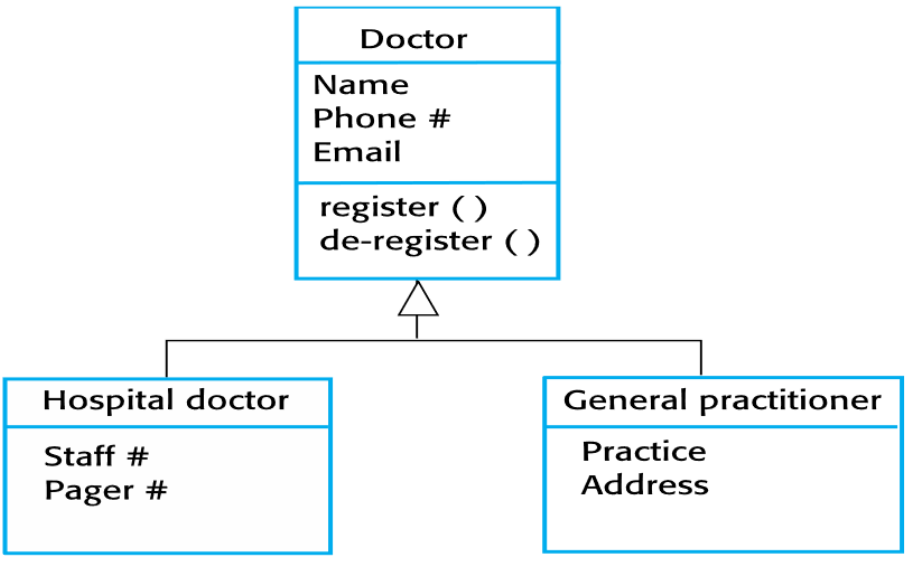
Generalization hierarchy

A generalization hierarchy



Generalization hierarchy with added detail

A generalization hierarchy with added detail



---

5. Behavioral Models

Behavioral_Models

- models of dynamic behavior of system while executing; show what happens or supposed to happen when system responds to stimulus from environment

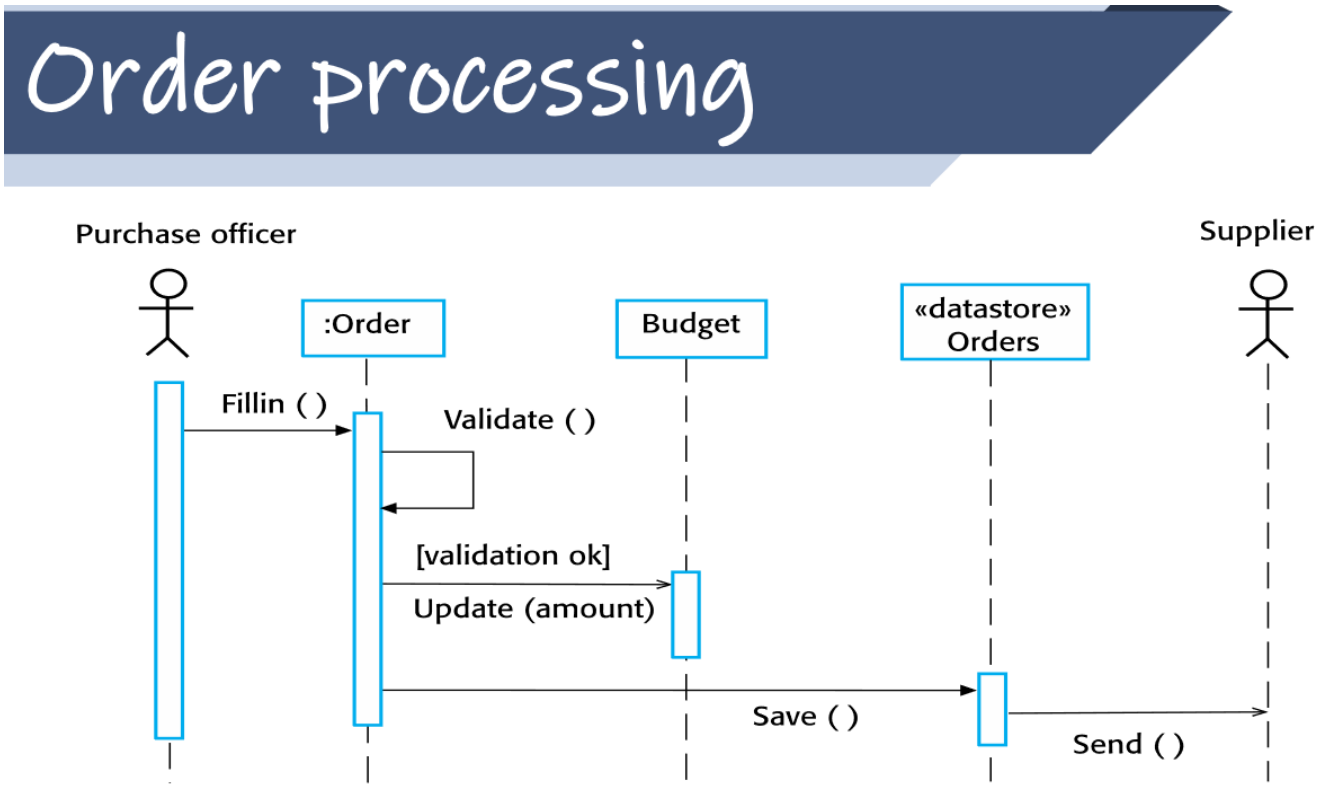
Two types of stimuli

- **Data**: to be processed by system
- **Events**: trigger system processing; may have associated data

Data-driven modeling

- many business systems are data-processing systems that are primarily driven by data; controlled by data input to system, with little external event processing
- show sequence of actions involved in processing input data and generating associated output
- particularly useful during analysis of requirements as they can be used to show end-to-end processing in system

Order processing



Event-driven modeling

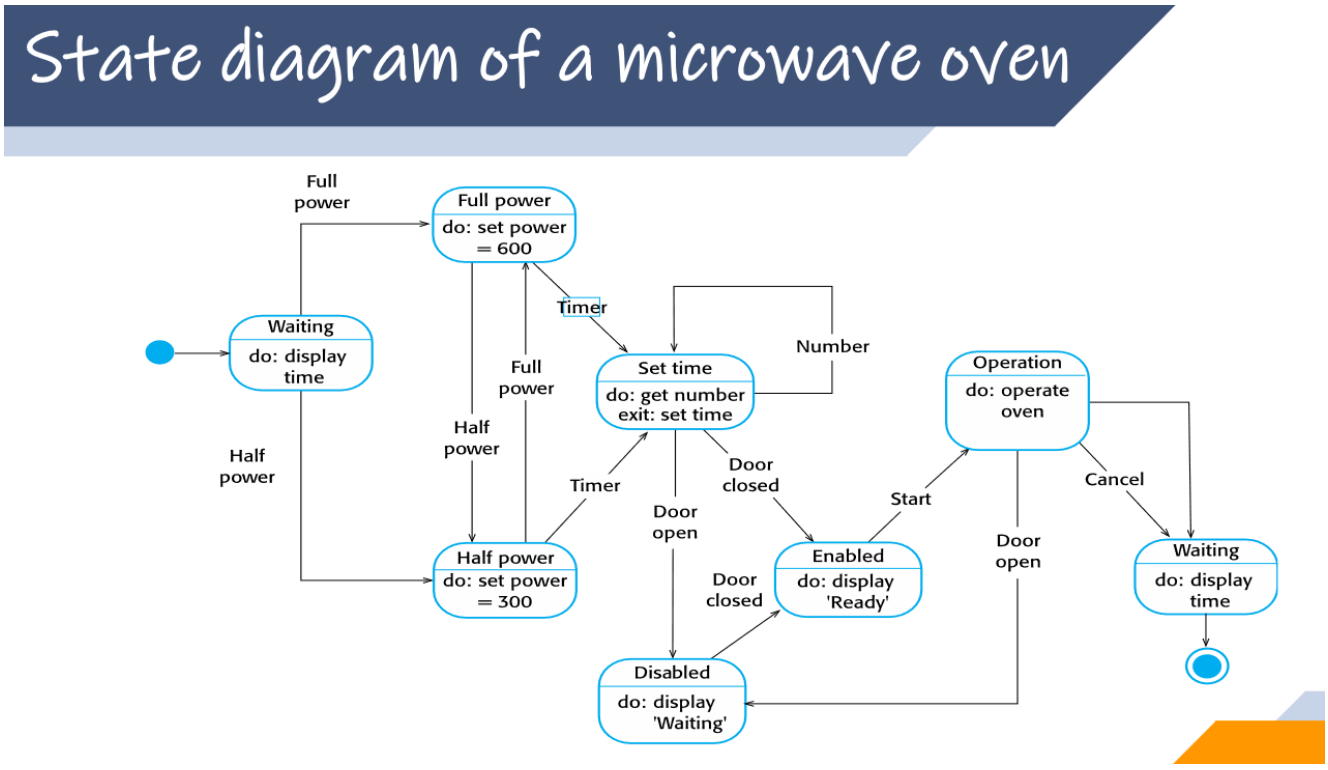
- real-time system are often event-driven, with minimal data processing (landline phone switching system responds to events such as receiver off hook by generating dial tone)
- event-driven modeling shows how system responds to external/internal events
- based on assumption that system has finite number of states and events (stimuli) may cause transition from one state to another

State machine models

- model behaviors of system in response to external/internal events
- show system responses to stimuli; often used for modeling real-time systems
- show how system states as nodes and events as arcs between nodes; when event occurs, system moves form one state to another
- **State charts** are integral part of UML and used to represent state machine models

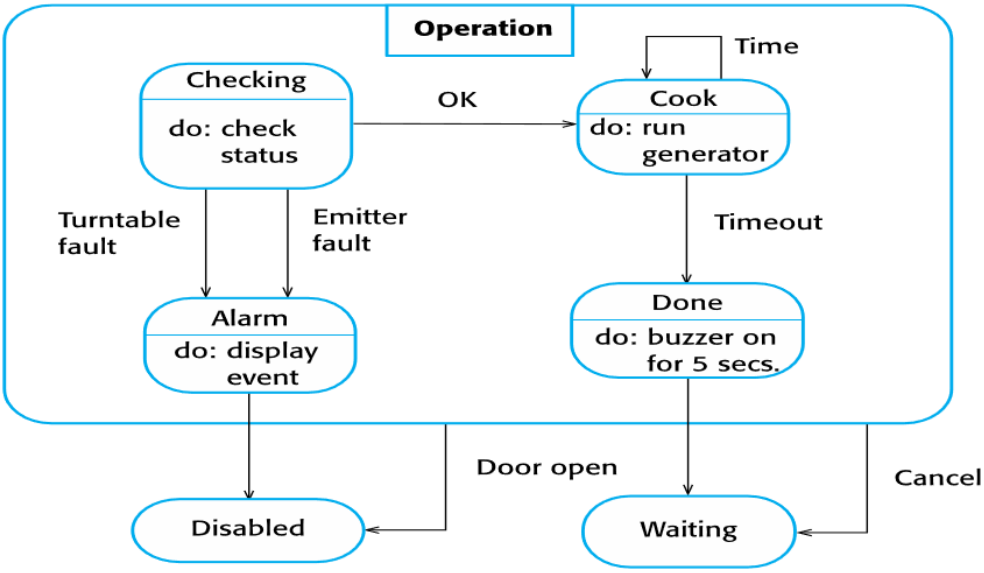
State diagrams

State diagram of microwave oven



Microwave oven operation

Microwave oven operation



---

6. Model-Driven Engineering (MDE)

Model-Driven_Engineering

- approach to software development where models rather than programs are principal outputs of development process
- programs that execute on hardware/software platform are then generated automatically from models
- proponents argue that this raises level of abstraction in software engineering so engineers no longer have to be concerned with programming language details or specifics of execution platforms

Usage of MDE

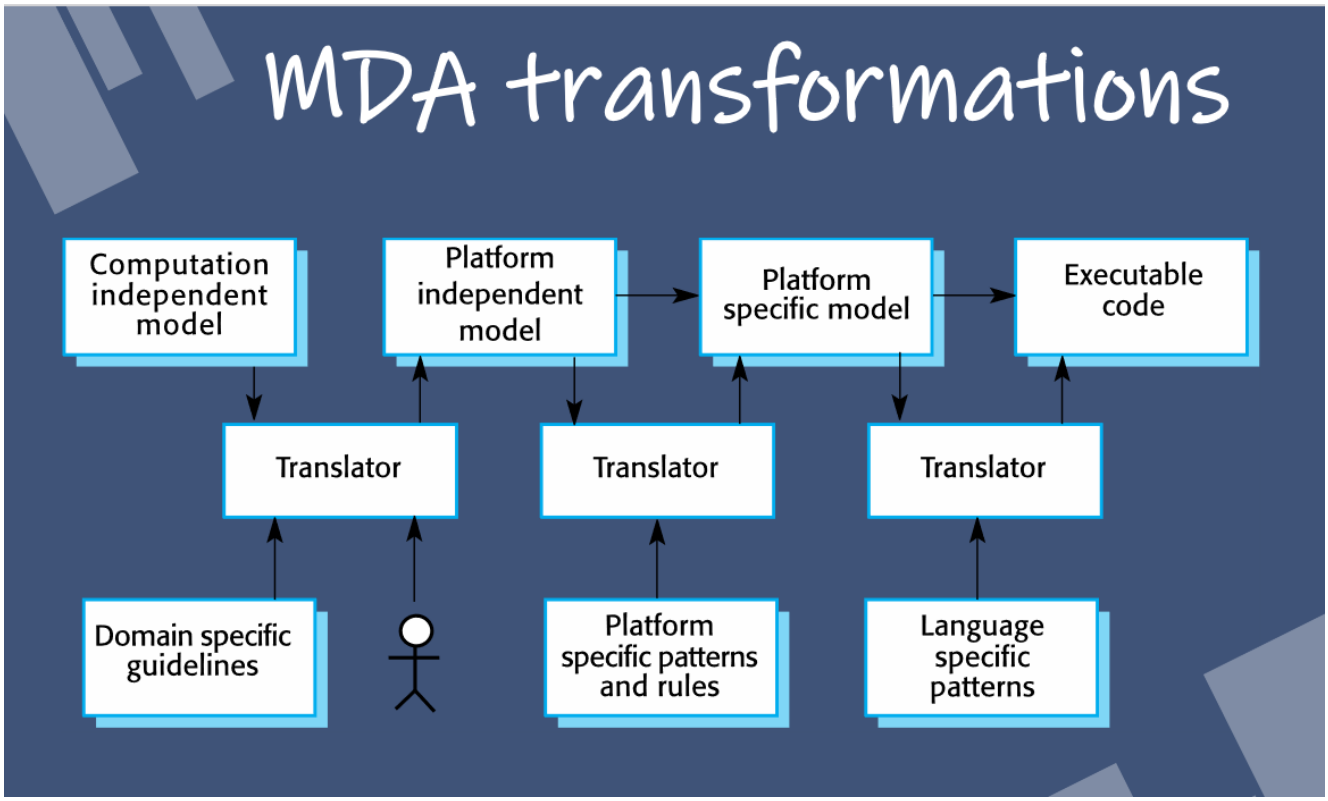
- early stage of development, and unclear whether/not will have significant effect on software engineering practice
Pros	Cons

| allow systems to be considered higher levels of abstraction | models for abstraction and not necessarily right for implementation |
| generating code automatically means it is cheaper to adapt systems to new platforms | savings from generating code may be outweighed by costs of developing translators for new platforms |

Types of model

1. **A computation independent model (CIM)**
   - important domain abstraction used in system; sometimes called domain models
2. **A platform independent model (PIM)**
   - model operation of system without reference to implementation; usually described using UML models showing static system structure and how it responds to external/internal events
3. **Platform specific models (PSM)**
   - transformation of platform-independent model with separate PSM for each application platform; there may be layers of PSM, with each layer adding some platform-specific detail

MDA transformations



Adoption of MDA

- range of factors limited adoption of MDE/MDA

- specialized tool support required to convert models from one level to another
- limited tool availability and organizations may require tool adaptation and customization to their environment
- for long-lifetime system developed using MDA, companies are reluctant to develop their own tools/rely on small companies that may go out of business
- models are good way of facilitating discussions about software design; however, abstractions useful for discussions may not be right abstractions for implementation
- for most complex systems, implementation is not the major problem- requirements engineering, security and dependability, interaction with legacy systems, and testing are more significant

---