System Modeling

Eunseok Lee, Prof.
School of Information and Communication
Sungkyunkwan University

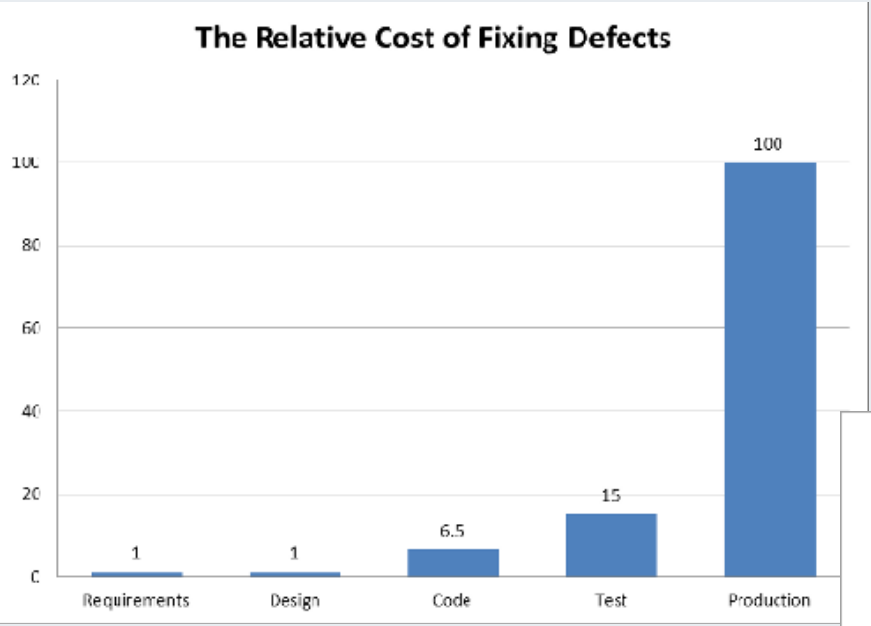
Objectives

- Understand how graphical models can be used to represent software systems;
- Understand why different types of model are required and the fundamental system modeling perspectives of context, interaction, structure, and behavior;
- Introduce some of the diagram types in the Unified Modeling Language (UML) and how these diagrams may be used in system modeling;
- Be aware of the ideas underlying model-driven engineering, where a system is automatically generated from structural and behavioral models.

Topics covered

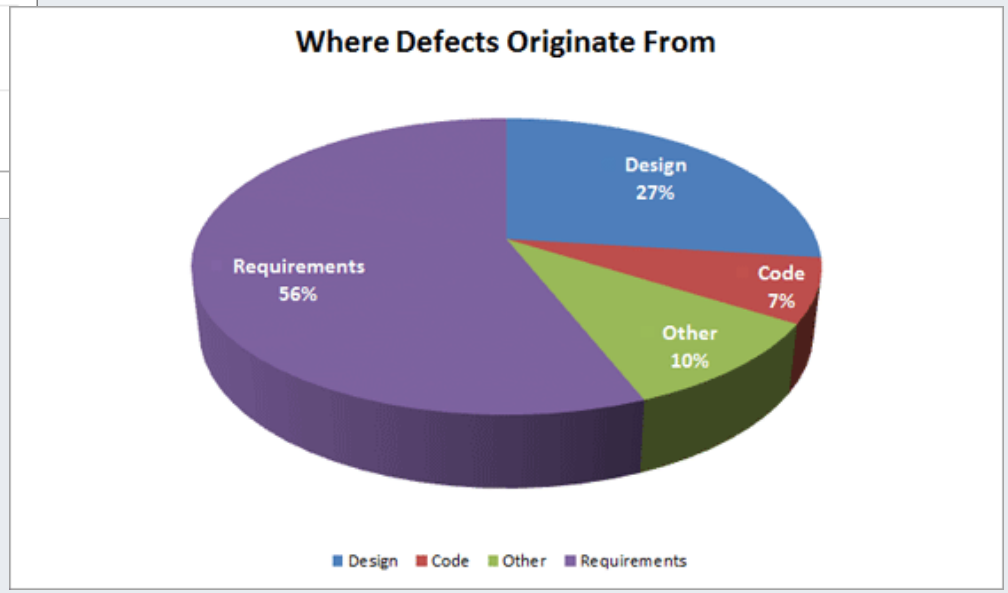
1. Context models
2. Interaction models
3. Structural models
4. Behavioral models
5. Model-driven engineering

Relative cost of fixing defects

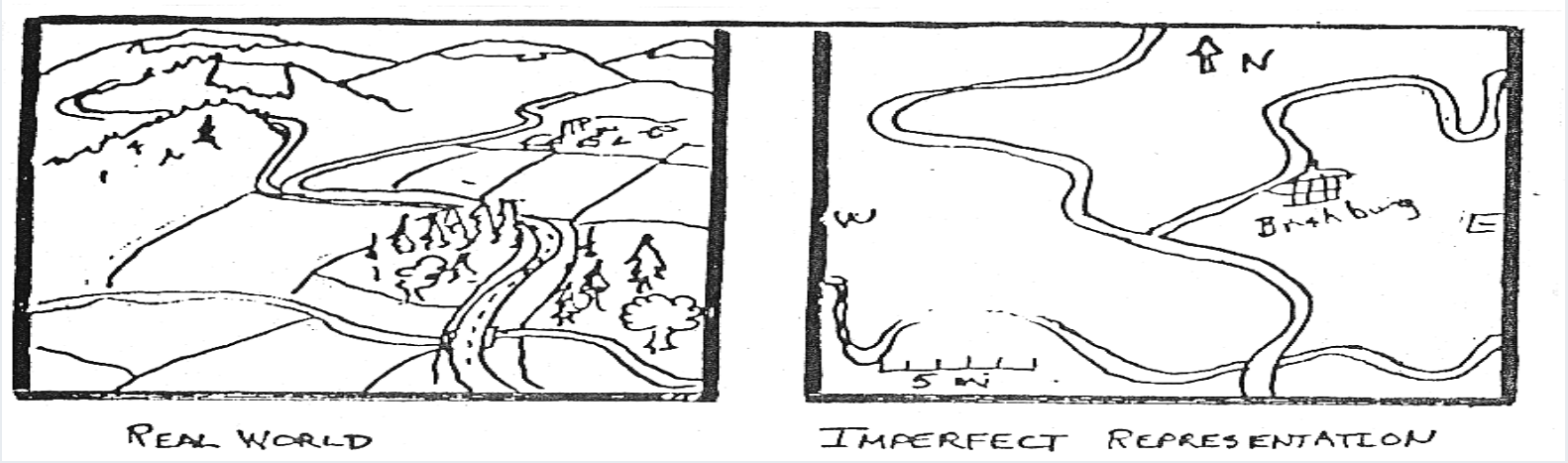


IBM Systems Sciences Institute

Mogyorodi, G. (2003) What is Requirements Based Testing? The Journal of Defense Software Engineering,



System modeling – what is the model?



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.

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.

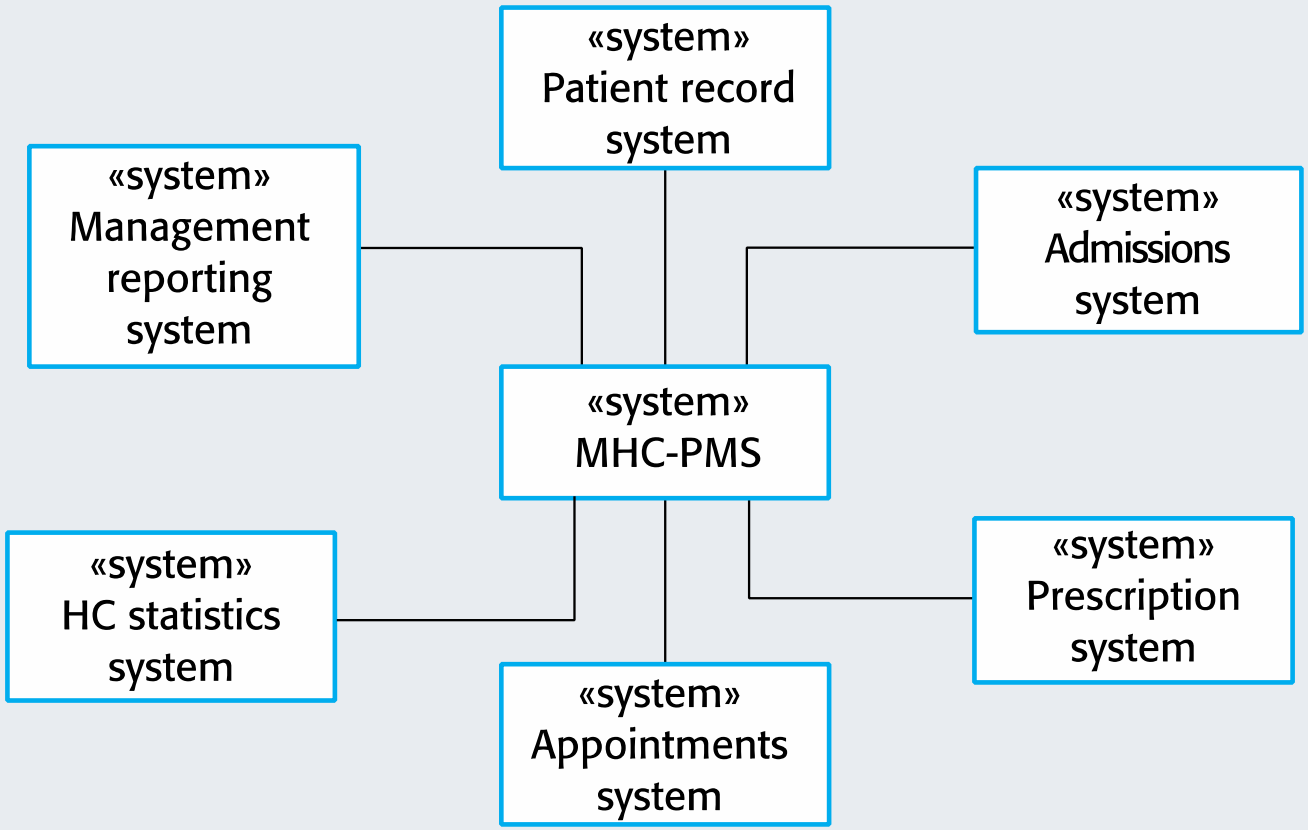
Use of graphical models

- **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*.

1. Context models

- Context models are used to illustrate the operational context of a system - they show what lies outside the system boundaries.
- 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.

The context of the MHC-PMS



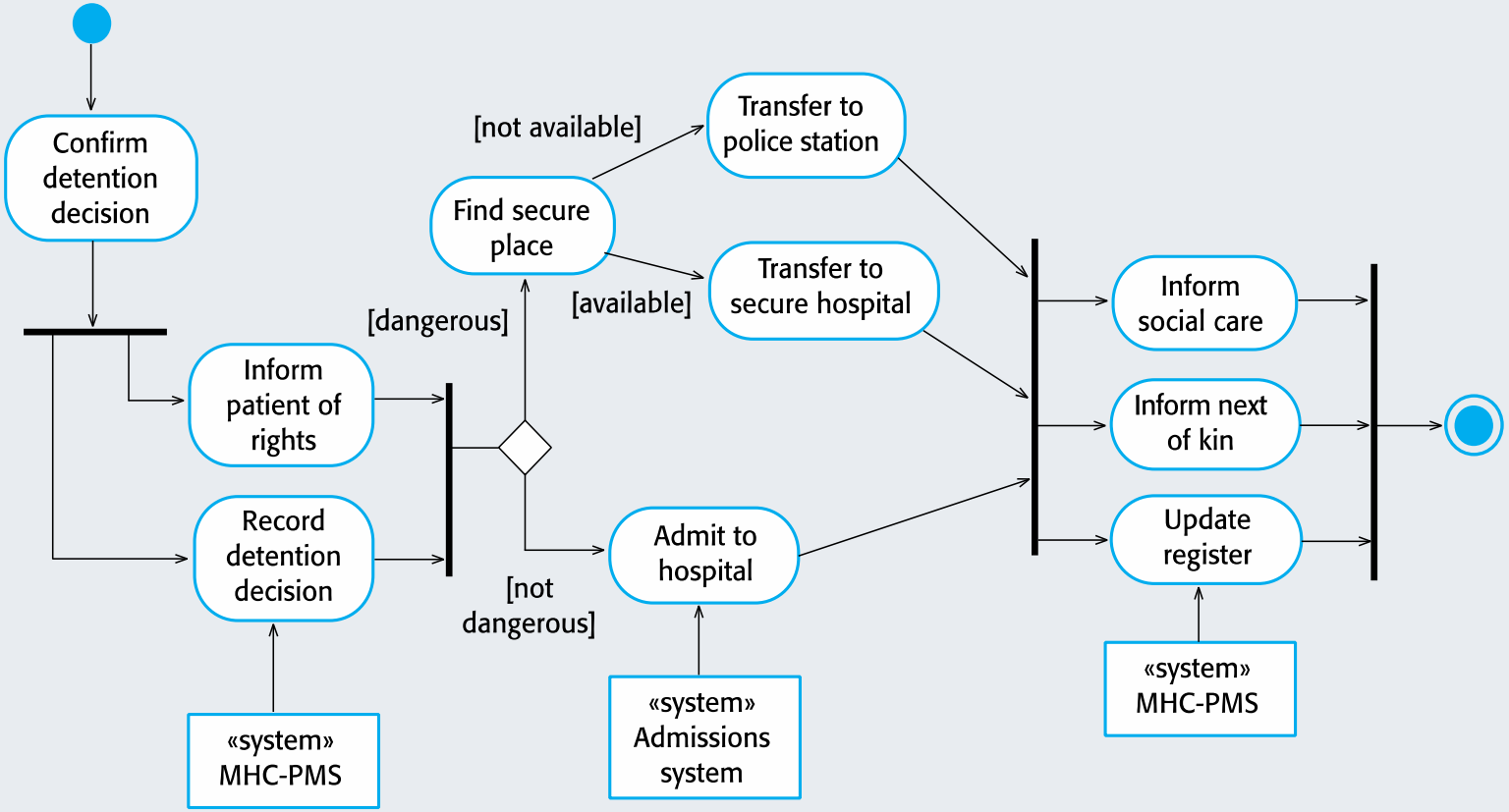
System boundaries

- **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.

1.1 Process models

- 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



2. Interaction models

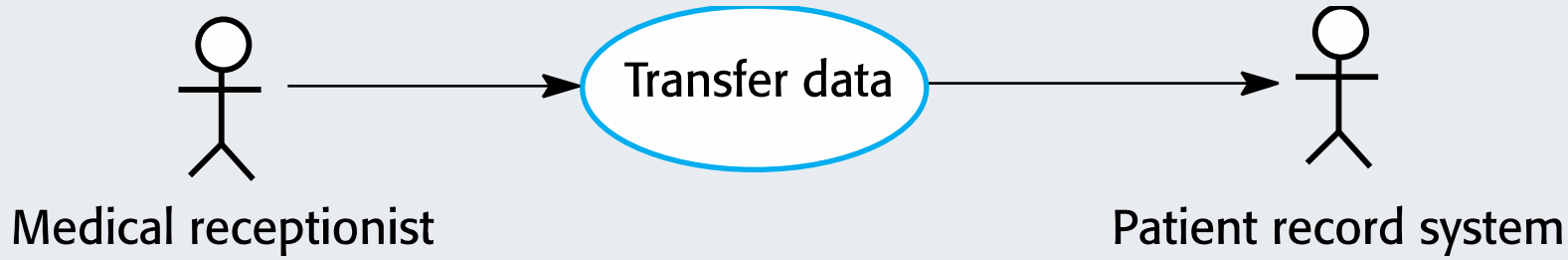
- Modeling **user interaction** is important as **it helps to identify user requirements**.
- Modeling **system-to-system interaction** highlights the **communication problems** that may arise.
- Modeling **component interaction** helps us understand if a proposed system structure is likely to deliver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

2.1 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 diagrammatically to provide an overview of the use case and in a more detailed textual form.

Transfer-data use case

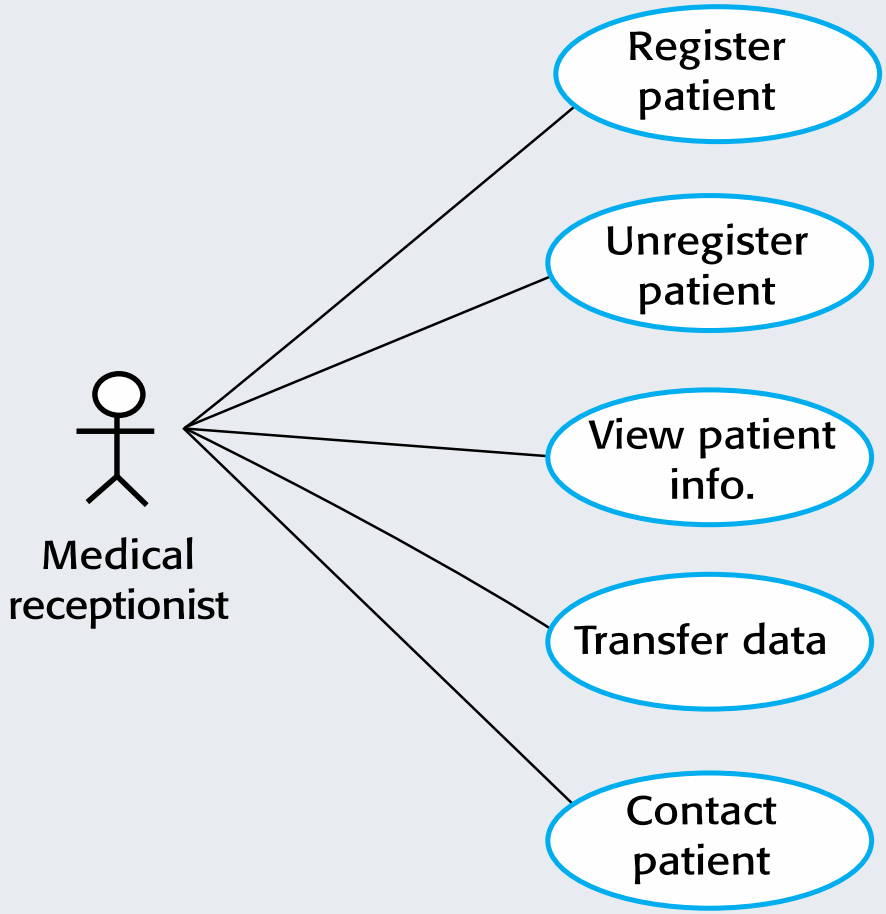
- A use case in the MHC-PMS



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 MHC-PMS 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.

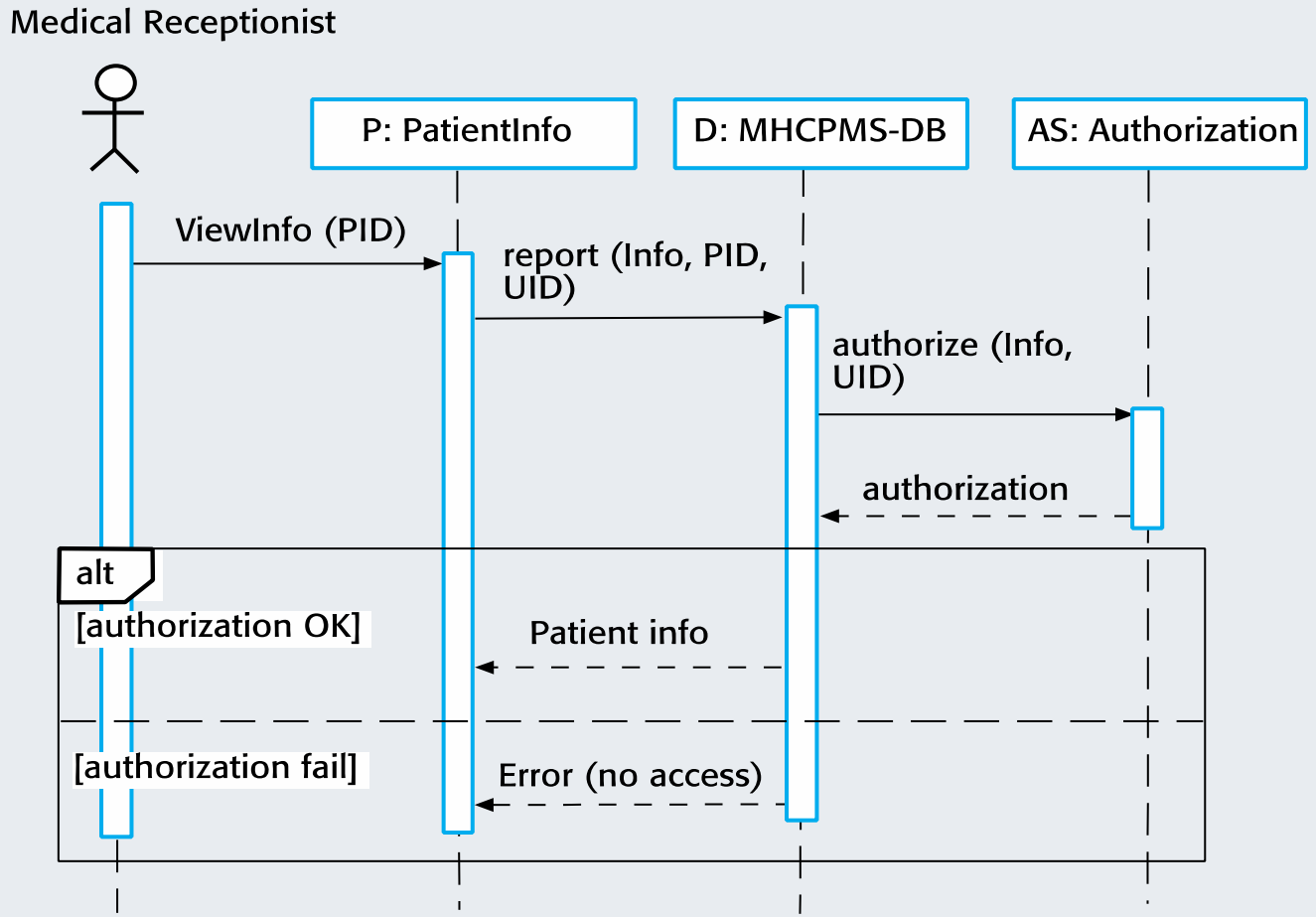
Use cases in the MHC-PMS involving the role 'Medical Receptionist'



2.2 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



3. Structural models

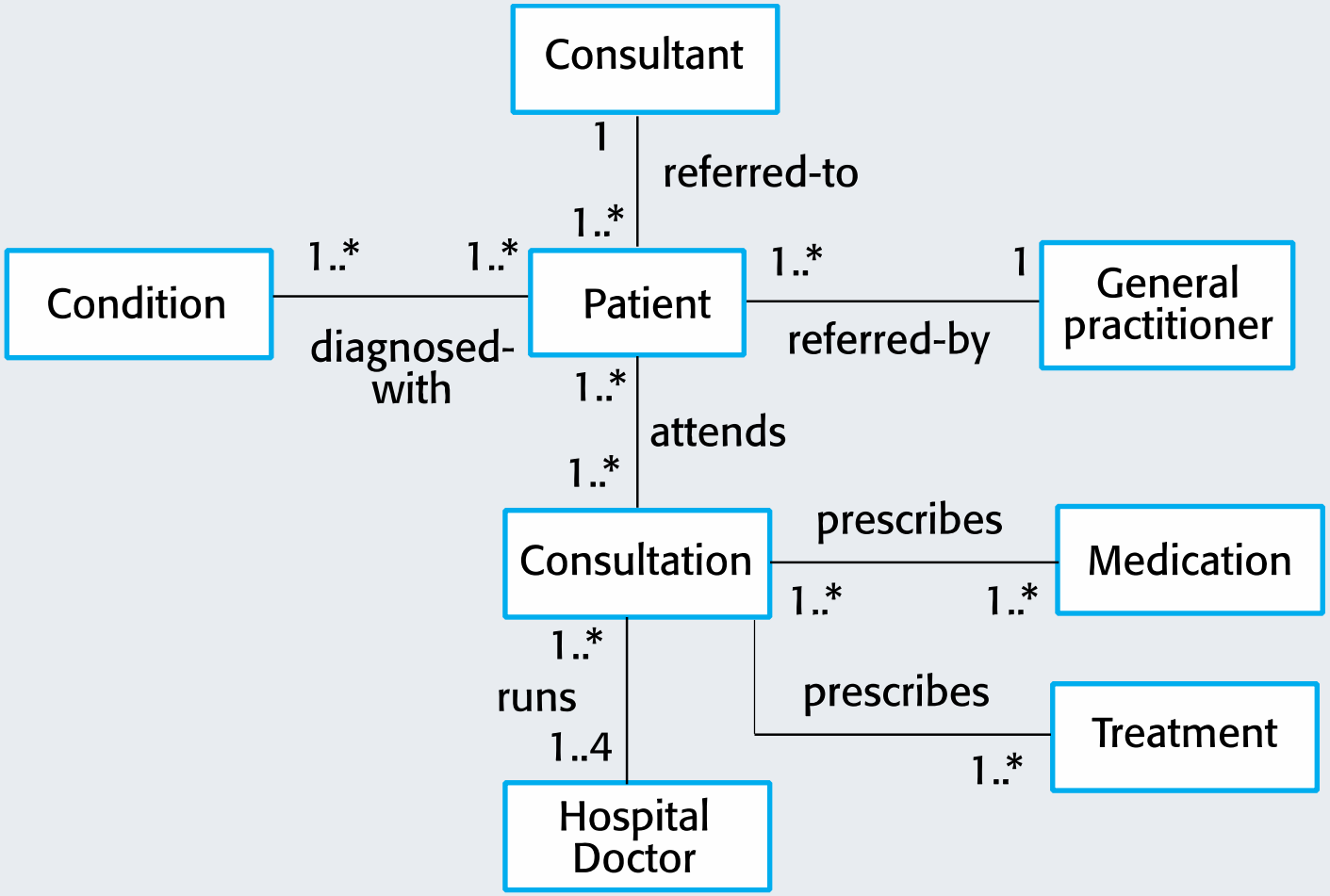
- Structural models of software display the organization of a system in terms of the *components* that make up that system and *their relationships*.
- Structural models may be static models, which show the structure of the system design, or dynamic models, which show the organization of the system when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

3.1 Class diagrams

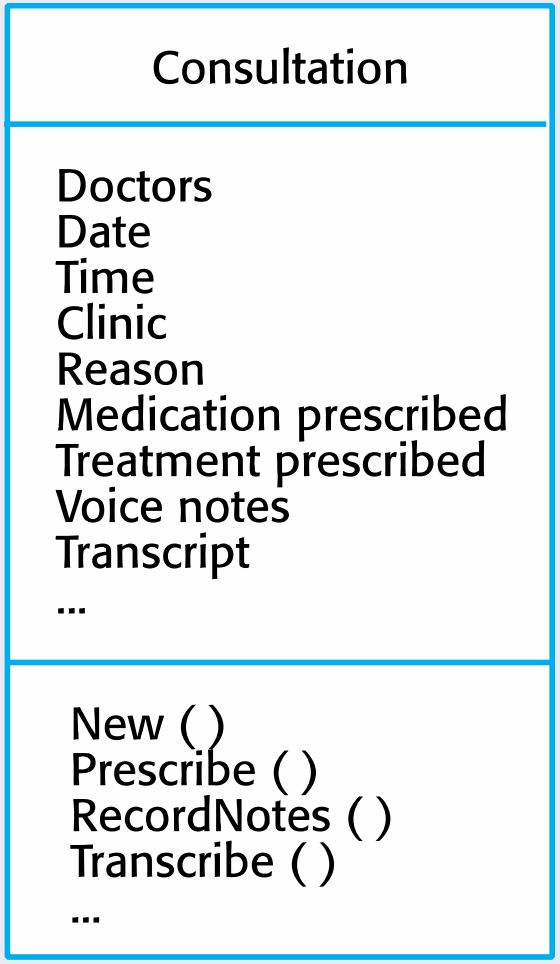
- Class diagrams are used when developing an object-oriented system model to show the *classes* in a system and the *associations* between these classes.
- A **class** can be thought of as a general definition of one kind of system object.
- An **association** is a link between classes that indicates 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



Key points

- A model is an abstract view of a system that ignores system details. Complementary system models can be developed to show the system's context, interactions, structure and behavior.
- Context models show how a system that is being modeled is positioned in an environment with other systems and processes.
- Use case diagrams and sequence diagrams are used to describe the interactions between users and systems in the system being designed. Use cases describe interactions between a *system* and *external actors*; sequence diagrams add more information to these by showing interactions between *system objects*.
- Structural models show the organization and architecture of a system. Class diagrams are used to define the static structure of classes in a system and their associations.

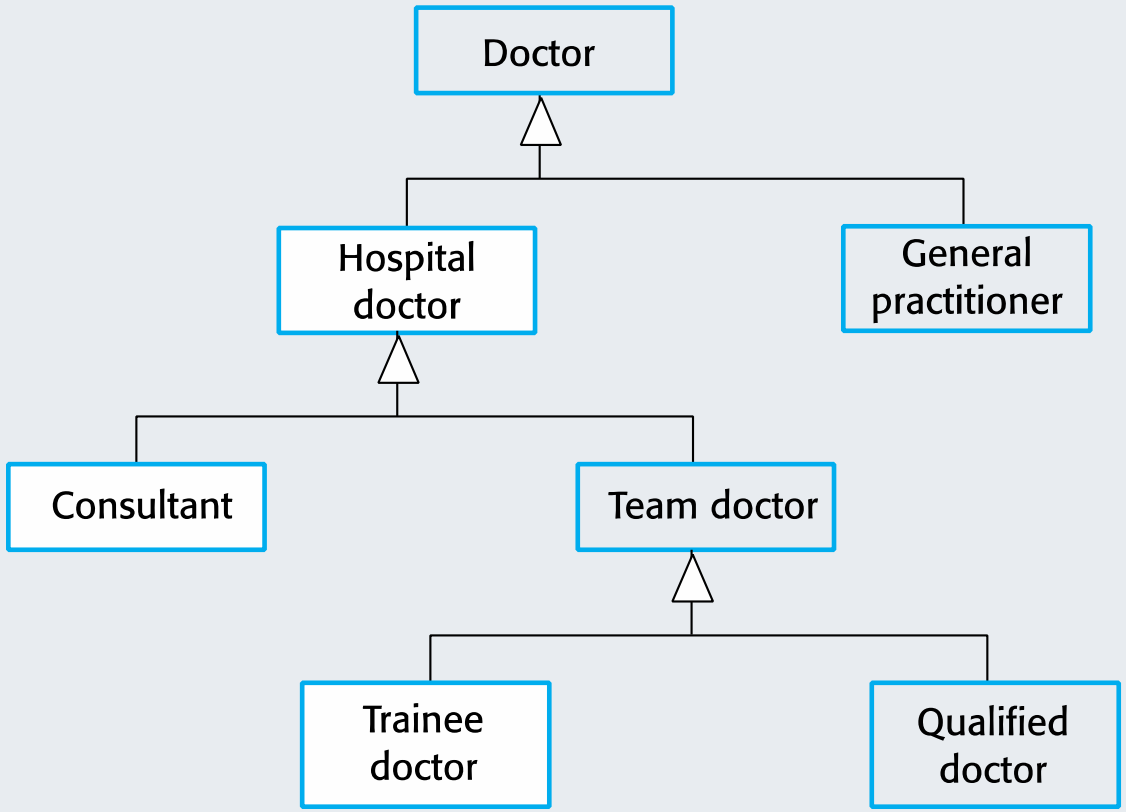
System Modeling

Part 2

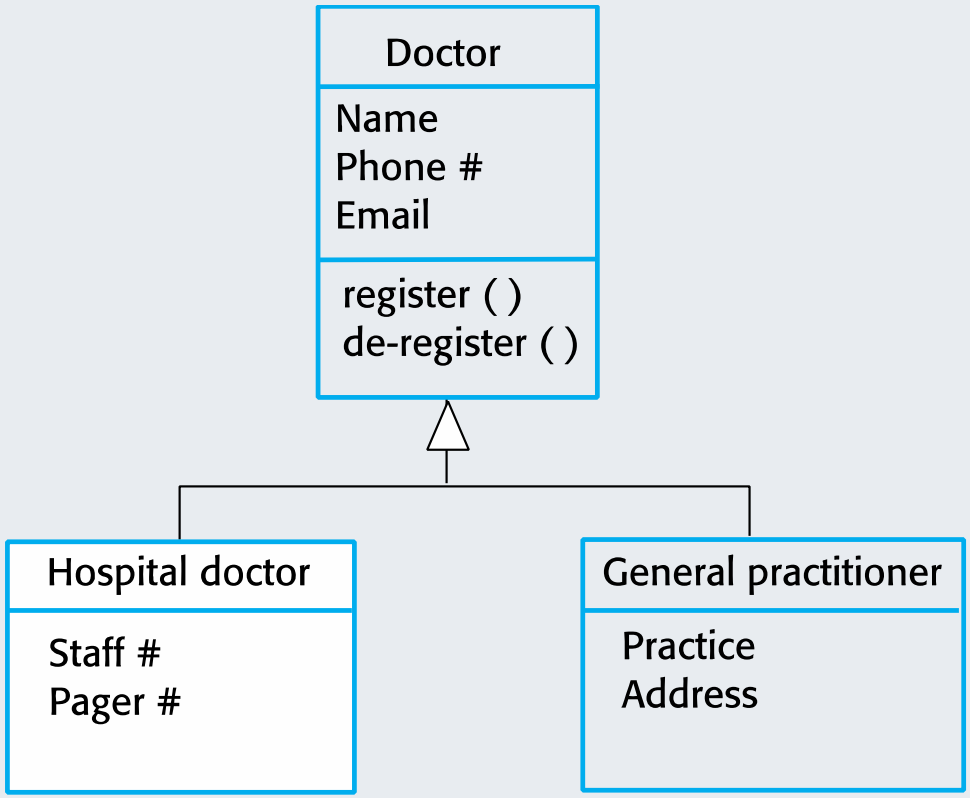
Generalization

- 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



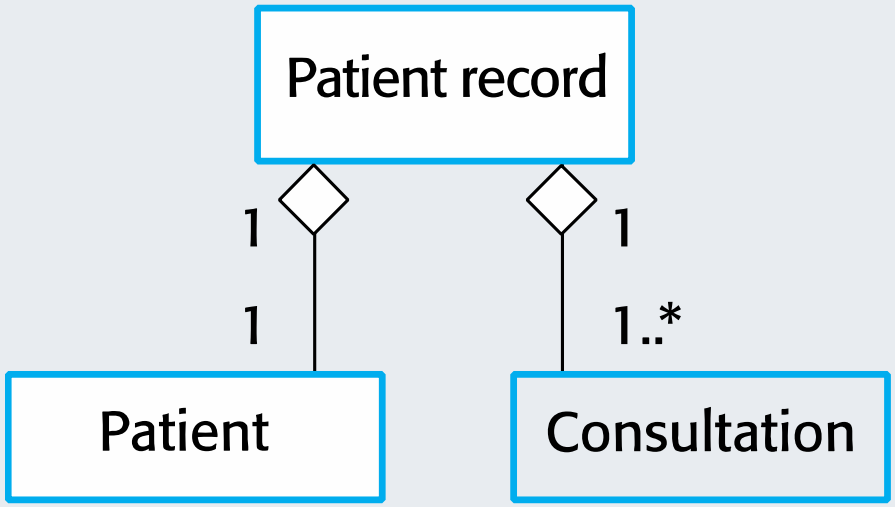
A generalization hierarchy with added detail



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



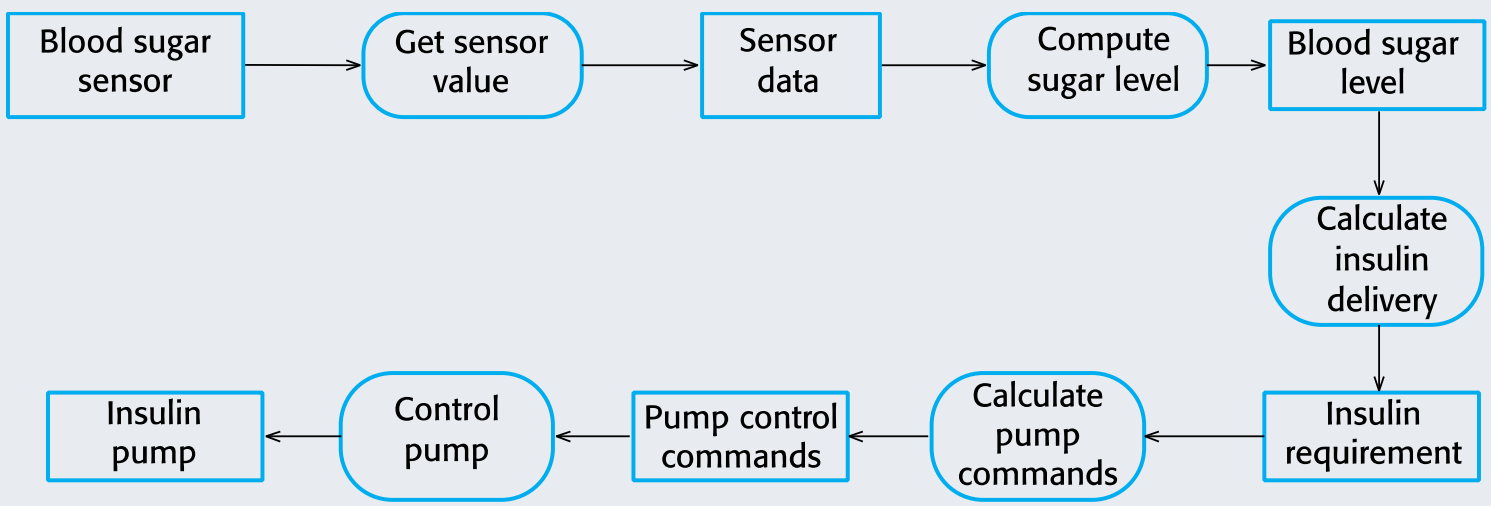
4. Behavioral models

- 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.

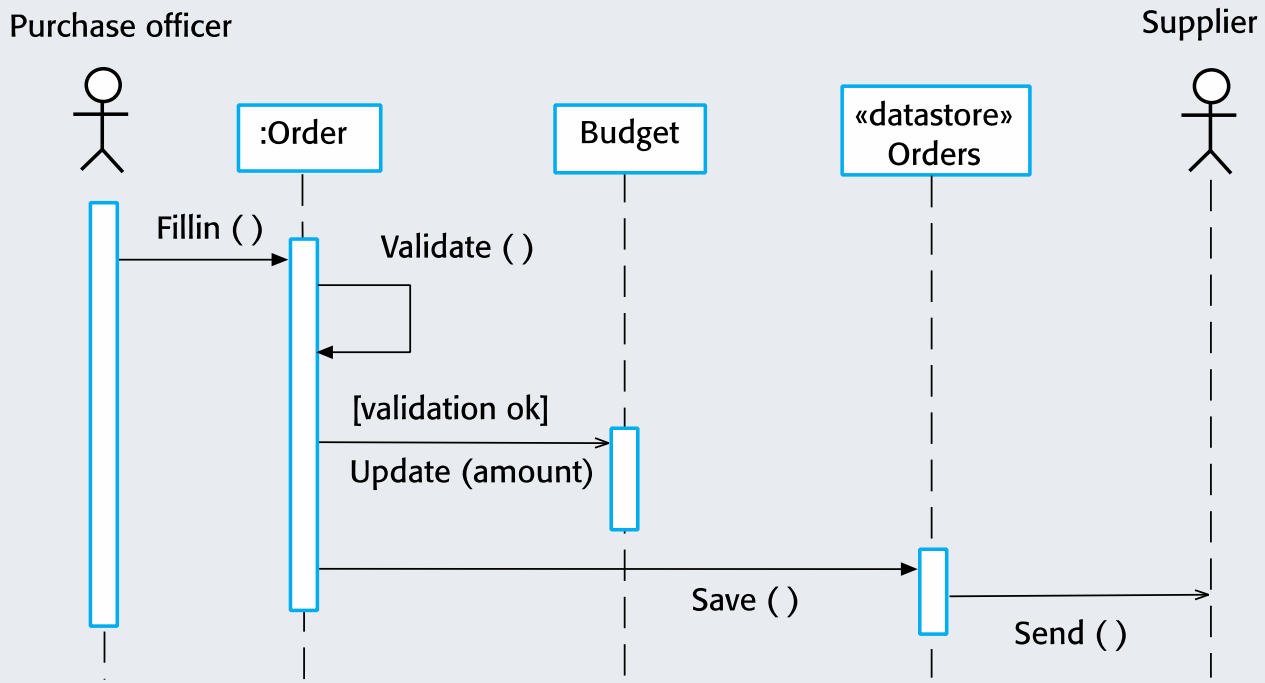
4.1 Data-driven modeling

- 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 in Sequence Diagram



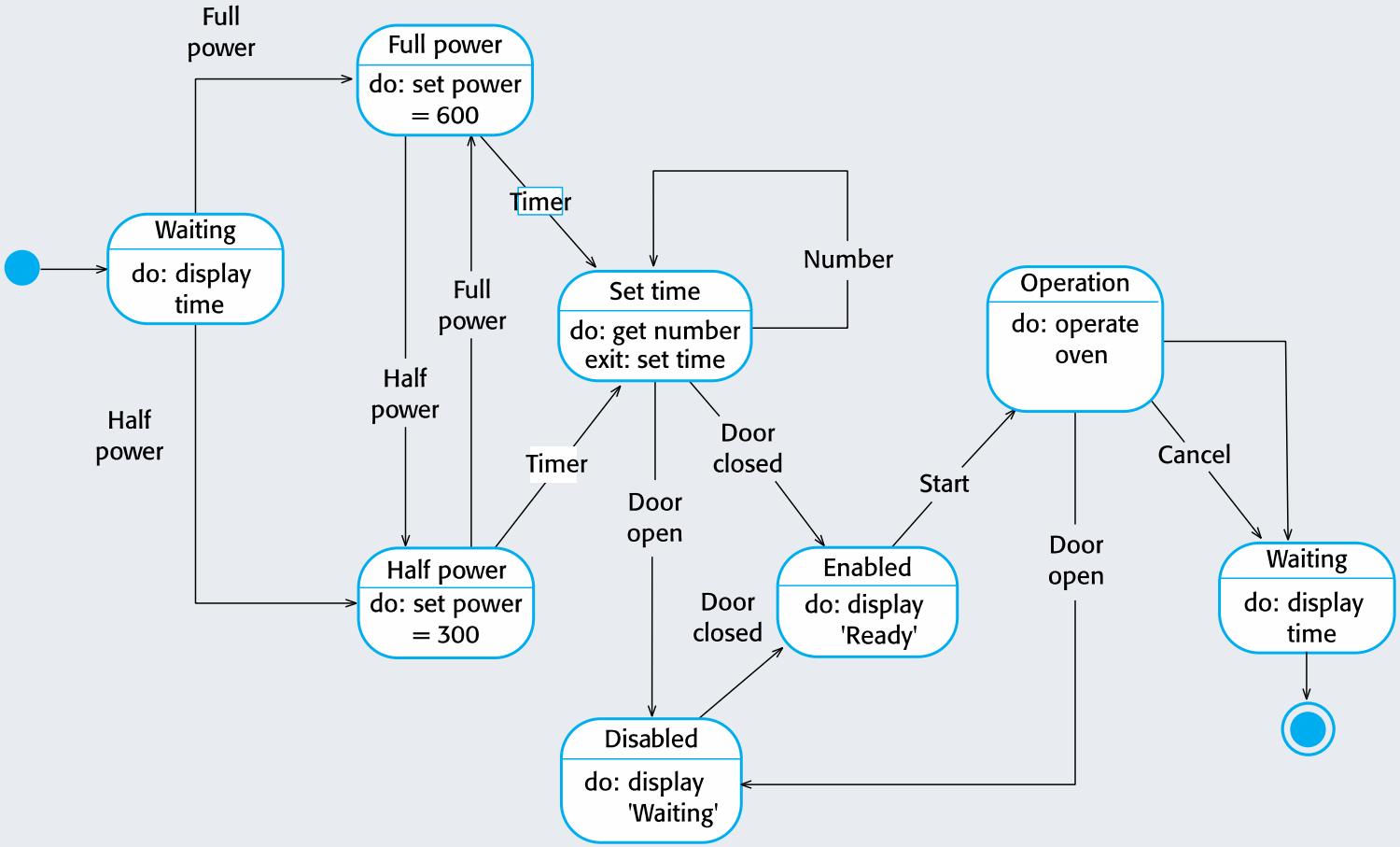
4.2 Event-driven modeling

- 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.

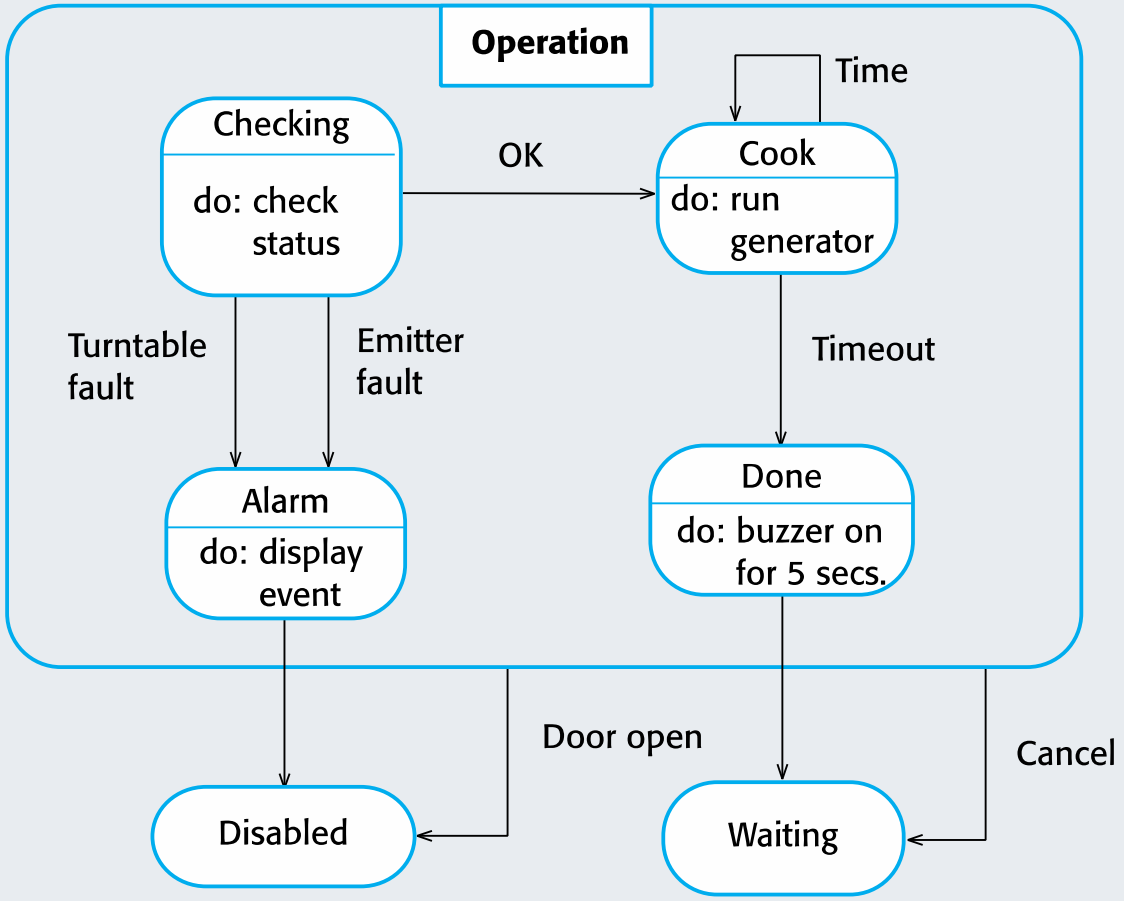
State machine models

- 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.

5. Model-driven engineering

- Model-driven engineering (MDE) is software development methodology where models rather than programs are the principal outputs of the development process.
- 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 so that engineers no longer have to be concerned with programming language details or the specifics of execution platforms.

Usage of model-driven engineering

- **Model-driven engineering is still at an early stage of development, and it is unclear whether or not it will have a significant effect on software engineering practice.**
- **Pros**
  - Allows systems to be considered at higher levels of abstraction
  - Generating code automatically means that it is cheaper to adapt systems to new platforms.
- **Cons**
  - Models for abstraction and not necessarily right for implementation.
  - Savings from generating code may be outweighed by the costs of developing translators for new platforms.

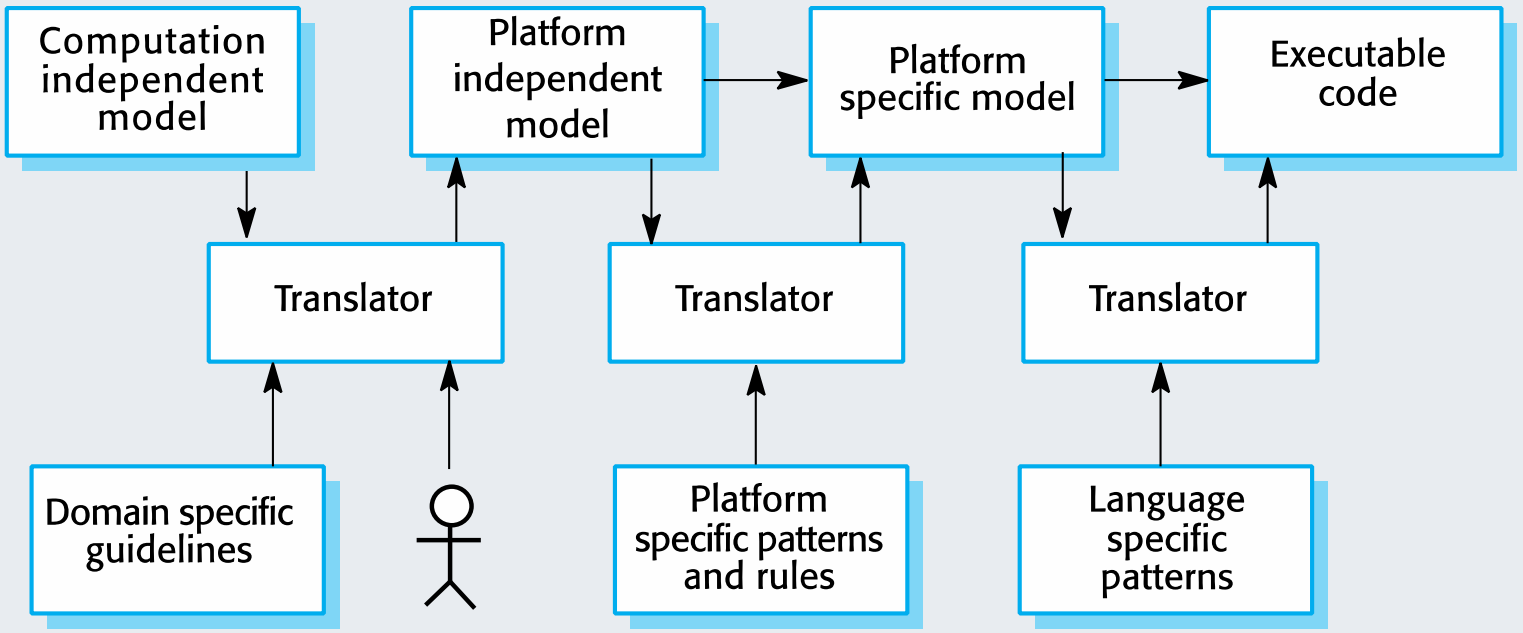
Model driven architecture

- Model-driven architecture (MDA®) was the precursor of more general model-driven engineering
- MDA is a model-focused approach to software design and implementation that uses a subset of UML models to describe a system.
- Models at different levels of abstraction are created. From a high-level, platform independent model, it is possible, in principle, to generate a working program without manual intervention.

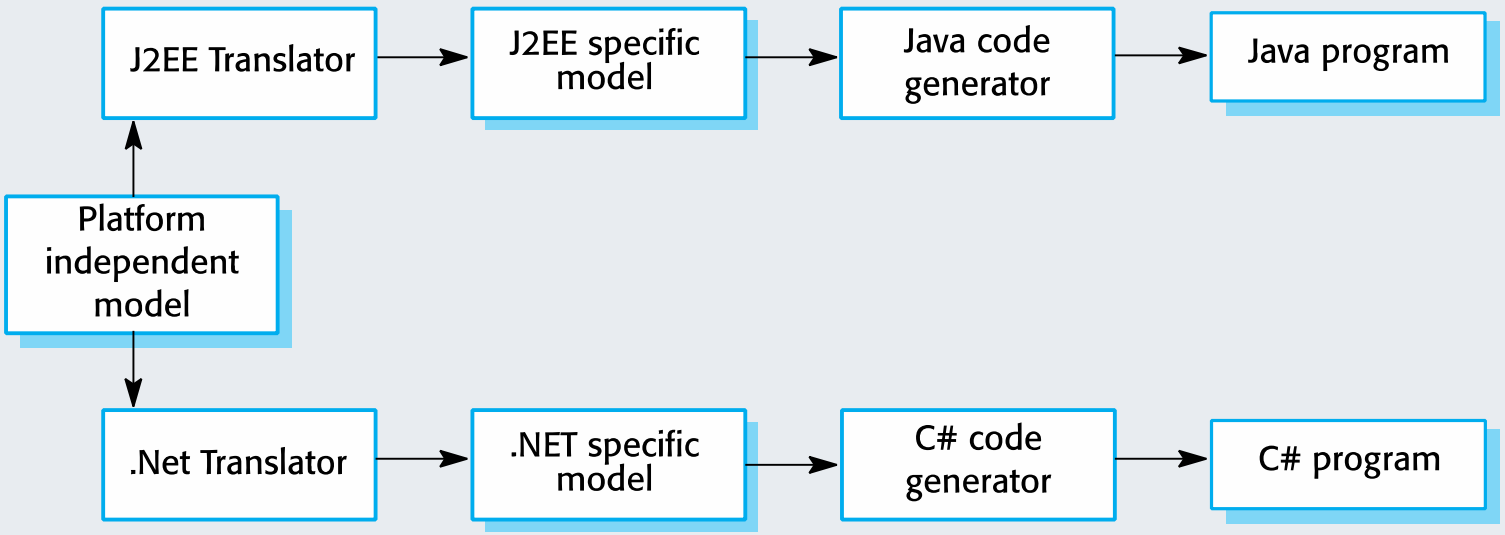
Types of model

- **A computation independent model (CIM)**
  - These model the important domain abstractions used in a system. CIMs are sometimes called domain models.
- **A platform independent model (PIM)**
  - These model the operation of the system without reference to its implementation. The PIM is usually described using UML models that show the *static system structure* and *how it responds* to external and internal events.
- **Platform specific models (PSM)**
  - These are transformations of the platform-independent model with a separate PSM for each application platform. In principle, there may be layers of PSM, with each layer adding some platform-specific detail.

MDA transformations



Multiple platform-specific models



Agile methods and MDA

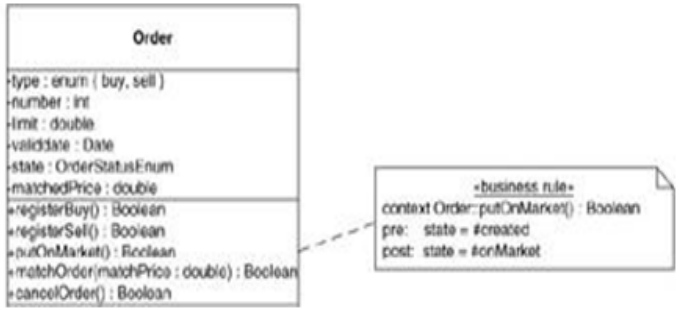
- The developers of MDA claim that it is intended to support an iterative approach to development and so can be used within agile methods.
- The notion of extensive up-front modeling contradicts the fundamental ideas in the agile manifesto and so suspect that few agile developers feel comfortable with model-driven engineering.
- If transformations can be completely automated and a complete program generated from a PIM, then, in principle, MDA could be used in an agile development process as no separate coding would be required.

Executable UML

- The fundamental notion behind model-driven engineering is that completely automated transformation of models to code should be possible.
- This is possible using a subset of UML2, called Executable UML or xUML.

Features of executable UML

- **To create an executable subset of UML, the number of model types has therefore been dramatically reduced to these 3 key types:**
  - **Domain models** that identify the principal concerns in a system. They are defined using *UML class diagrams* and include objects, attributes and associations.
  - **Class models** in which classes are defined, along with their attributes and operations.
  - **State models** in which a *state diagram* is associated with each class and is used to describe the life cycle of the class.
- **The dynamic behavior of the system may be specified declaratively using the object constraint language (OCL), or may be expressed using UML's action lang**



Key points

- Behavioral models are used to describe the dynamic behavior of an executing system. This behavior can be modeled from the perspective of the data processed by the system, or by the events that stimulate responses from a system.
- Activity diagrams may be used to model the processing of data, where each activity represents one process step.
- State diagrams are used to model a system's behavior in response to internal or external events.
- Model-driven engineering is an approach to software development in which a system is represented as *a set of models* that can be automatically transformed to executable code.

LG에 한방 먹은 SK

카카오뱅크 수주 경쟁서
리눅스 앞세운 LG CNS 승리
SK 금융권 사업 타격 불가피

국내 첫 인터넷전문은행 시스템 구축 경쟁에서 LG CNS가 SK주식회사를 제쳤다. 기술력과 혁신성에서 LG CNS가 SK를 앞섰다는 평가다.

7일 정보기술(IT) 업계에 따르면 인터넷전문은행 카카오뱅크는 최근 IT시스템 구축 우선협상 대상자로 LG CNS를 선정했다. 이번 시스템 입찰에는 금융IT 분야 라이벌 격인 LG CNS와 SK주식회사 C&C 부문이 참여해 치열한 경쟁을 벌였다. LG CNS가 경쟁에서 승리하면서 금융 분야에서 주도권을 갖게 될 것으로 예상된다.

업계 관계자는 "오는 15일까지 우선협상 기간으로 특별한 문제점이 없으면 LG CNS가 시스템 구축 사업자로 선정된 것으로 봐도 무방하다"고 말했다.

카카오뱅크는 지난 2월 시스템통합(SI) 업체를 대상으로 시스템 구축을 위한 사업제안서를 발송했다. 수주 경쟁에는 LG CNS, SK주식회사 C&C 부문 등이 참여했다. LG CNS는 은행권 최초 리눅스 운영체제(OS) 도입과 독자 보유 기술인 모델 기반 개발방식(MDD)을 제안했고, SK주식회사는 기존 은행권 시스템인 유닉스 체제로 입찰에 참여한 것으로 알려졌다.

업계는 리눅스 기반 시스템을 제안한 LG CNS 혁신성에 카카오뱅크가 큰 점수를 준 것으로 평가했다. 리눅스는 은행권 전산 시스템으로 자리 잡고 있는 IBM 메인프레임이나 최근 도입된 유닉스보다 개방성, 호환성 등이 뛰어나고 비용 부담이 덜하다는 장점이 있지만 안정성이 검증되지 않은 게 단점으로 지적돼 왔다. 이 때문에 보수적인 금융권에서는 리눅스 도입을 꺼려왔던 게 사실이다. 반면 뉴욕증권거래소를 비롯해 메릴린치 등 외국에서는 2000년대 초부터 리눅스를 쓰고 있다.

LG CNS는 "국내에서도 지난해 6월 한국거래소(KRX) 차세대 시스템으로 리눅스가 도입돼 안정성을 인정받았다"며 "이미 여러 성공 사례가 있는 만큼 국내 은행권에서도 무리 없이 적용할 수 있을 것"으로 자신했다. 리눅스와 함께 시스템 개발 기간과 비용을 20% 정도 절감할 수 있는 MDD 방식도 크게 어필했던 것으로 알려졌다. LG CNS는 "카카오뱅크와 협의해 당초 계획대로 올 하반기 출범할 수 있도록 할 것"이라고 말했다.

SK주식회사는 국내 첫 인터넷은행 시스템 구축 경쟁에서 탈락함에 따라 금융권 사업에서 타격이 불가피할 것으로 보인다. 회사 관계자는 "은행 전산 시스템에서 리눅스가 실현된다면 혁신적"이라며 "LG CNS 성과를 지켜보면서 리눅스 도입 여부를 고려하겠다"고 말했다. 오찬종 기자

LG CNS와 SK C&C 최근 수주 현황

LG CNS	SK C&C
농협, 광주은행, 전북은행, 카카오뱅크(우선협상 중)	국민은행(태블릿 브랜치), 우리은행(일부), 부산은행

핵심전자소자 10분의 1 비용으로 만든다

지질硏·고려대 연구팀 개발

국내 연구진이 휘어지는 디스플레이와 반도체 등 전자산업 핵심으로 불리는 '박막 트랜지스터'를 기존 공정의 10분의 1 수준으로 값싸게 제작할 수 있는 기술을 개발했다

있었다. 진공증착 대신 용액으로 박막 트랜지스터를 만들면 이런 문제가 해결되지만 전기가 잘 통하지 않는 문제가 생겼다.

연구진은 '리간드' 공정을 활용해 이 문제를 해결했다. 박막 트랜지스터를 만드는 기판 위에 은과 산화알루미늄 등을 떨어트리고 빠르게 회전시키면 원심력에 의해 얇