Software Engineering

(CS3201)

System Models

Avinash

Mahindra™ University

Global Thinkers. Engaged Leaders.

ÉCOLE CENTRALE SCHOOL OF ENGINEERING

System Modelling

Introduction

- System Models are something that may be developed as part of Requirements Engineering (though they may also be used in other processes, especially in design phase)
- System Models: Graphical models (diagrams), which can be used to represent Software systems
- System Modelling Perspectives (how you want to view system):
  - Context
  - Interaction
  - Structure
  - Behavior
- UML provides a set of diagrams that can be used to represent system models

Introduction

- One more facet to System Modelling – Model driven engineering
- Model driven engineering: Executable system is generated from structural and behavioral models

System Modelling

- System modeling: Process of developing abstract models of a system
- "abstract model": Simplified representation of the system, process or data
- There are different kinds of abstract models, each differ in the kind of view or perspective it presents
- System modelling, in common usage, means representing the system in graphical notation based on diagram types in the UML (however, there can be formal mathematical models as well – not covered at this point in the course)

Why (and where to) use models?

- In Requirements Engineering: To help find/ understand detailed requirements for a system
- In Design Process: Describe the system to engineers implementing the system
- After Implementation: To document the system structure and operation

What do you usually model?

- Existing Systems
- Systems to be developed

- Existing systems models:
  - Used in Requirements Engineering to clarify what existing system does
  - Focus stakeholder discussion on strengths and weaknesses
- Models of systems to be developed:
  - Explain the proposed requirements to the stakeholders
  - Engineers use these models to discuss design and document for implementation
  - Model driven Engineering process: Generate complete or partial system from models

What is / is NOT a system model

- System Model is NOT a complete representation of the system
- Purposely leaves out detail making it easier to understand
- It is an **abstraction of the system**, rather than a representation of the system
- An abstraction is a deliberate simplification of the system, picking up the most salient features
- Sommerville: "Slides are an abstraction of the textbook. An English to Italian translation of the book is an alternate representation"

Models and Perspectives

- You may make different models of the system to represent different perspectives:
    - External perspective: Models the context or environment of the system
    - Interaction perspective: Models the interactions between a system and its environment, or between the components of a system
    - Structural perspective: Models the organization of a system or the structure of the data processed by the system
    - Behavioral perspective: Models the dynamic behavior of the system and how it responds to events

Three common usages of graphical models

- To stimulate and focus discussion about an existing or proposed system
- As a way of documenting an existing system
- In model-based engineering processes, as a detailed system description that can be used to generate a system implementation (here the models have to be complete)

Modelling and UML

- UML has 13 different types of diagrams to support creation of different kinds of system models
- Class diagrams: Shows the system structure (by showing the classes/ objects in the system and their association/ dependency)
- Use case diagrams: Interaction between system and environment
- Sequence diagrams: Interaction between actors and system and system components
- Activity diagrams: Activities involved in a process or in data processing
- State diagrams: System response to internal and external events

Various System Models – Context, Interaction, et al.

Context Models

- At early stage of system specification, you need to decide on system boundaries
- System boundaries? What is, and is not, part of the system
- Involves working with stakeholders to decide what functionality should be included within a system, and what processing and operations should be carried out by the system's environment
- This decision should be made early in the process to limit system cost and time needed to understand the system requirements and design

System Boundary?!

- In simple terms, A **system boundary** in a context model defines what is **inside** and **outside** the system you are designing. It separates the system from external elements (environment) like users, other systems, and hardware that interact with it
- Basically, the system boundary helps **clearly define what the system is responsible for** and what it depends on from the outside
- Example 1: Food delivery app
  - System boundary includes:
    - The app itself (ordering, payments, tracking)
    - The database storing restaurant and user information
  - Outside system boundary:
    - External services like Google Maps (though the app may interact with it)
    - Delivery drivers (they are external users)

System Boundary (boundaries are important)

- Example 2: Student portal
- **System boundary** defines what is **inside** the portal (what the system controls) and what is **outside** (external systems it interacts with)
  - Inside boundary (what system controls)
    - Student login & authentication
    - Course registration & schedules
    - Viewing grades & transcripts
    - Fee payment system
  - Outside boundary (external to system / outside system responsibility)
    - Email system (separately managed – by Microsoft outlook)
    - External payment gateways (e.g., UPI, bank systems)
- The system boundary helps <u>define what the university portal is responsible for</u> versus what it just connects with but doesn't control

Context Models

- In some cases, the boundaries between system and environment are clear
- Sometimes, the system boundaries are not made explicit (and can be configured to have different boundaries for different users) **
- Sometimes, system boundaries maybe defined by non-technical factors (social and organizational factors) ***
- Context models normally show the system and the environment (which may contain other systems). However, they do not show the types of relationships between the systems in the environment and the system that is being specified

** - Refer slide 21

*** - Refer slide 21

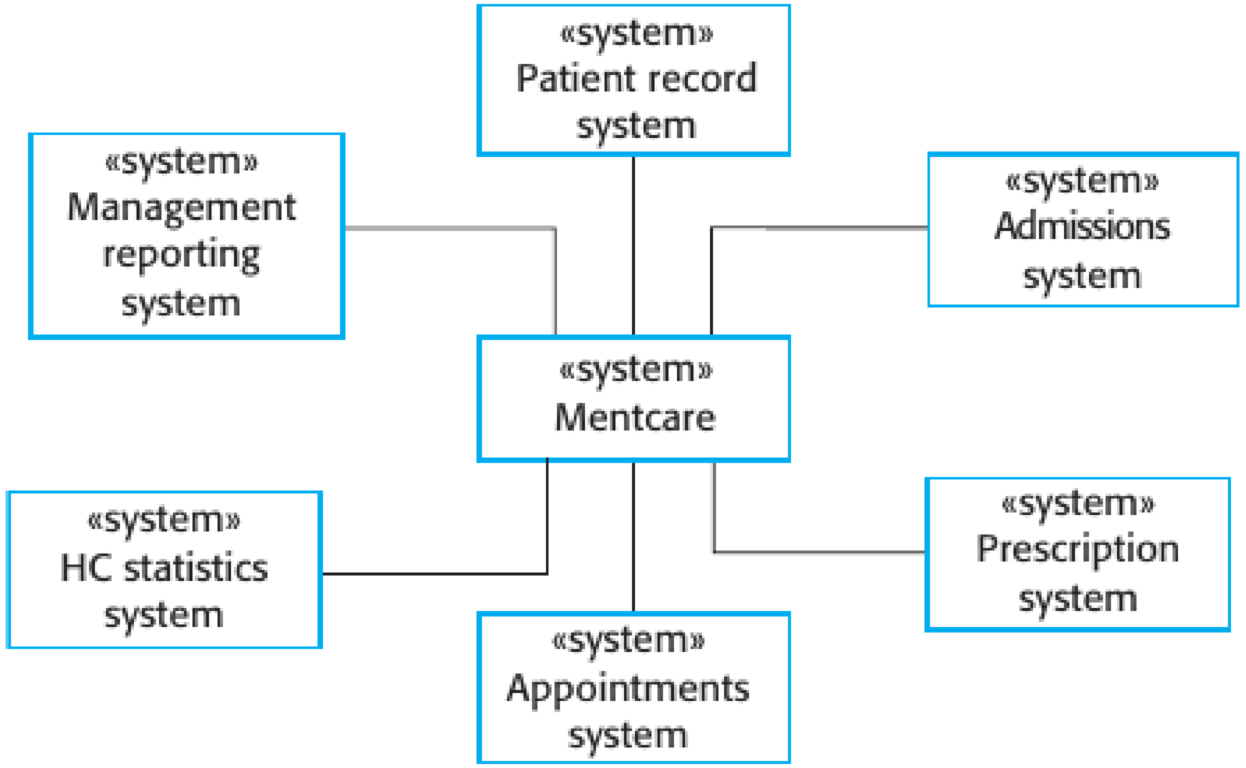
Context Models

- However, it might be important to know relationships between system and other systems in environment (external systems might produce or consume data for our system, direct or indirect network connections, etc.)
- These relationships are important, and can be captured using **business process models**
- UML activity diagrams may be used to show the business processes

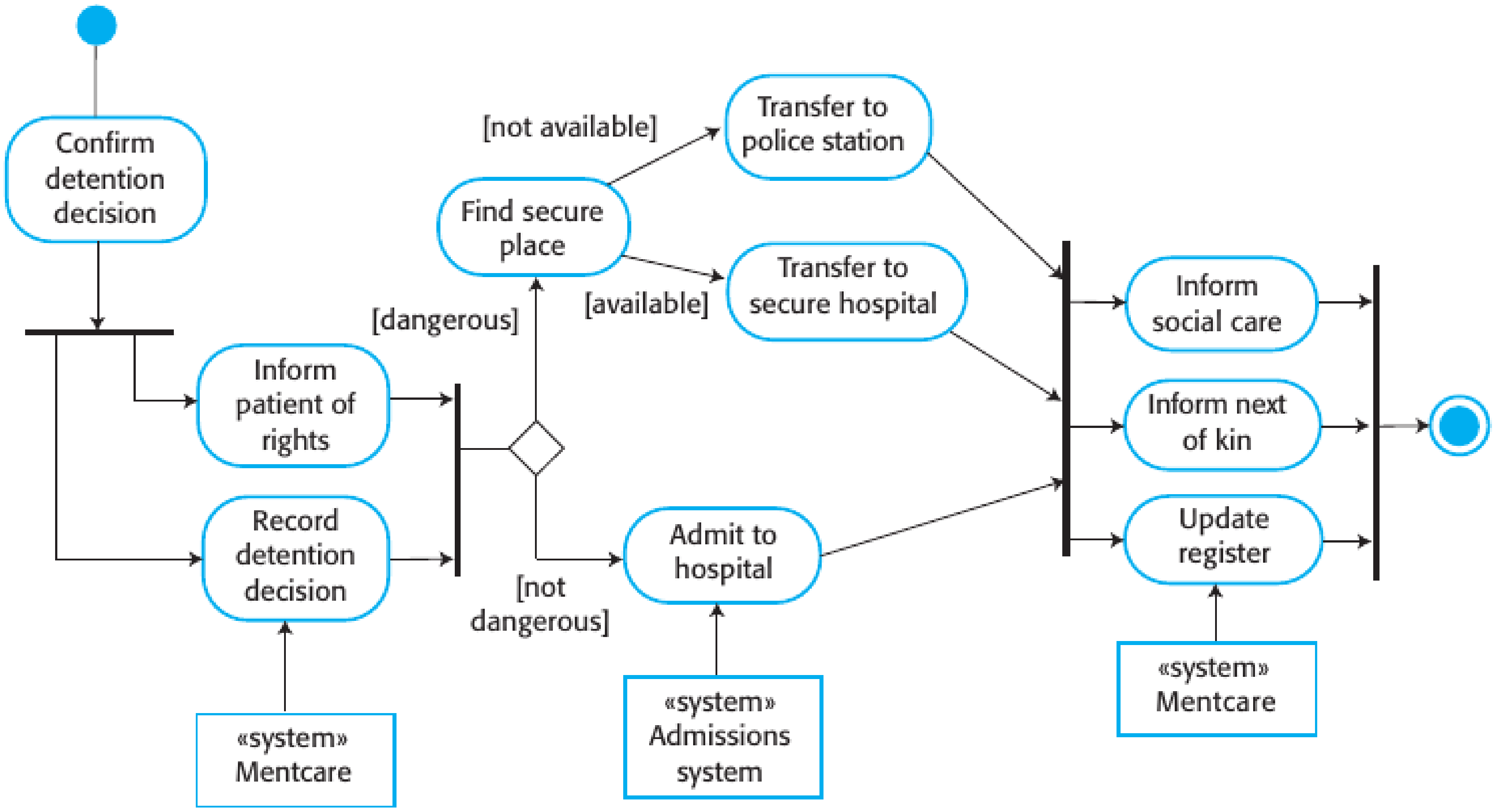
Business Process Models

- A **Business Process Model (BPM)** is a visual way of showing **how a business process works step by step**. It helps understand the flow of tasks, decisions, and interactions between people and systems
- Generally represented by UML Activity diagram (you will see this in your next SE lab Assignment 6)

Context Model: Example (SOM., pg. 142)


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

Business Process Model: Example (SOM., pg. 143)



Final thoughts – Context Models

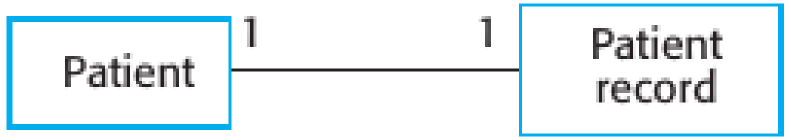
- ** - System boundaries are not always rigid and are based on how users interact with the system. In a university portal (system), the system does not include <taxation> module for the student, whereas the professor has a <taxation> module within the system boundary
- *** - Technology alone does not determine scope (boundary) of the system. Non-technical factors can play a role in defining boundary. The system does not just depend (and include) what it can do technically, but also covers non-technical areas. Example: Finance dept processes fees manually, but some might include it inside Juno to show that it is part of the overall system of Juno (which is responsible for all student matters)

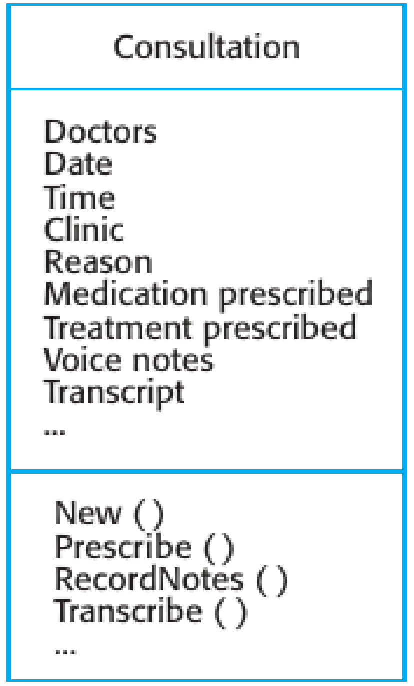
Structural Models

- "Structural models of software display the organization of a system in terms of the components that make up that system and their relationships" (SOM., pg. 149)
- Structural models can be static OR dynamic
  - Static: Shows organization of system (its design, components that make it)
  - Dynamic: Organization of a system while executing (a snapshot)

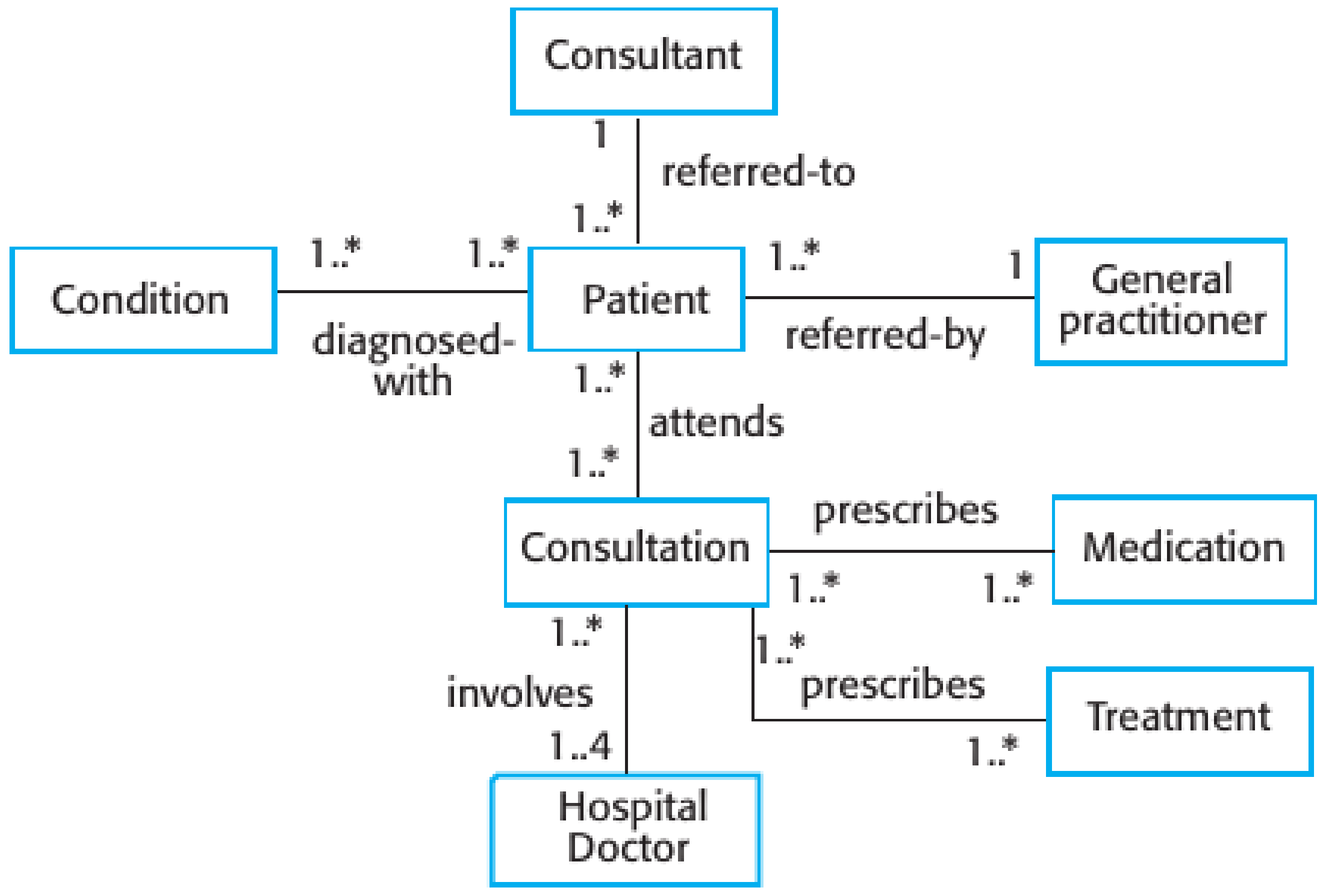
Class Diagrams

- "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 association is a link between classes indicating that some relationship exists between these classes."
- Class diagrams will be covered in depth in SE lab

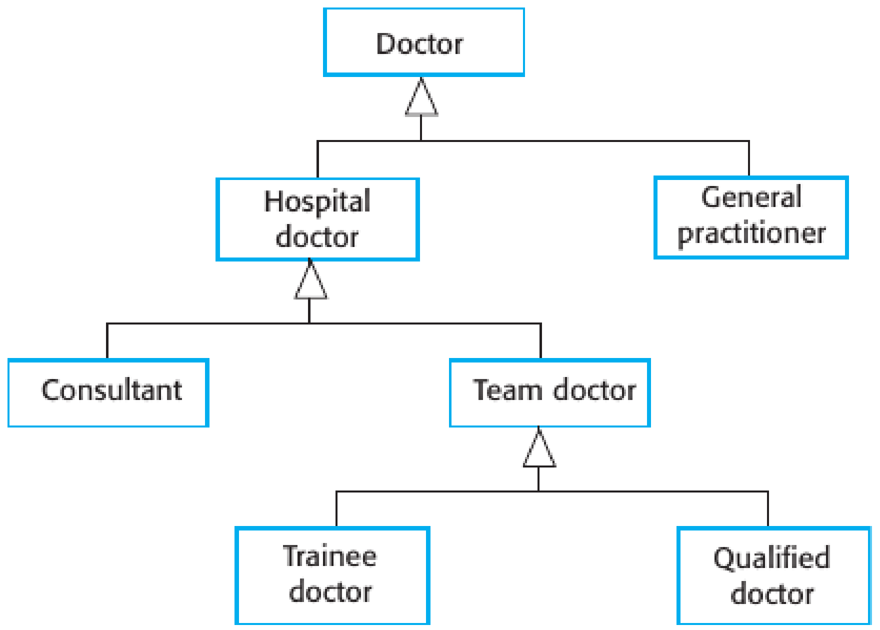


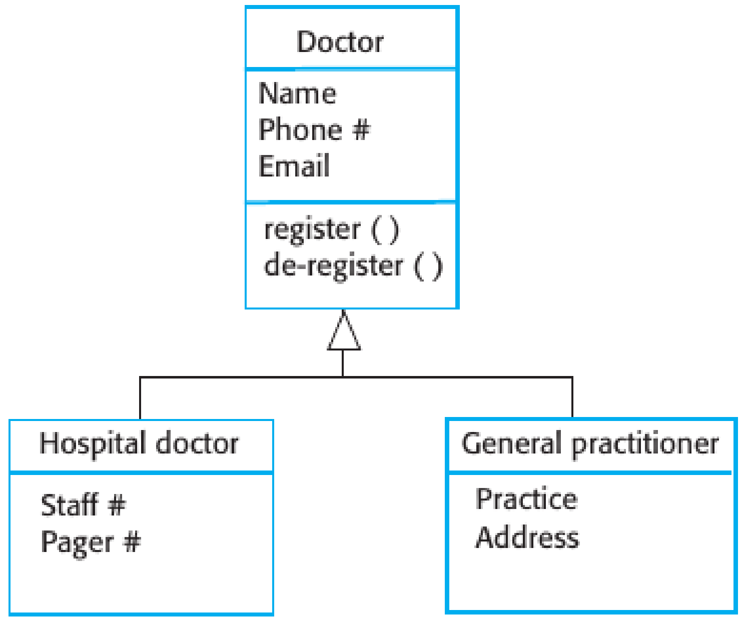


Classes and Associations in Mentcare (SOM., pg. 150)

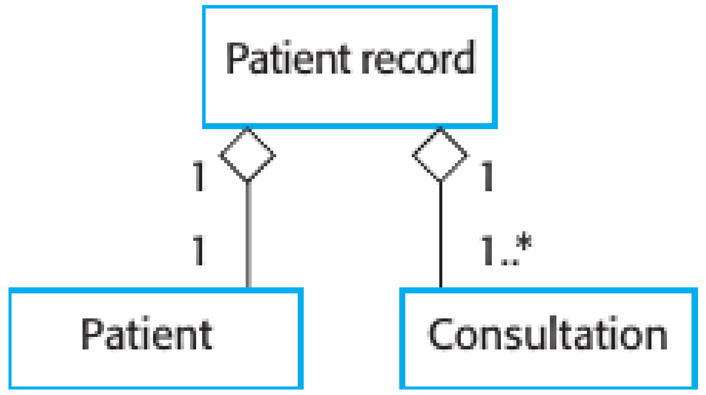


Generalization





Aggregation



Pop Quiz (assuming prior knowledge of OOP)

- How do you decide on what becomes a class in a class diagram?