Thakur Educational Trust's (Regd.)

THAKUR RAMNARAYAN COLLEGE OF ARTS & COMMERCE

ISO 21001:2018 Certified

PROGRAMME: B.Sc (I.T)

CLASS: S.Y.B.Sc (I.T)

SUBJECT NAME: SOFTWARE ENGINEERING

SEMESTER: IV

FACULTY NAME: Ms. SMRITI DUBEY

UNIT II

Chapter 4 – System Models

Concepts:

System Models and its types

Context Model

Interaction Model

Behavioral Model

Structured Model

Data Model

Introduction to System models

System modeling is the process of developing abstract models of a system, with each model presenting a different view or perspective of that system. It is about representing a system using some kind of graphical notation, which is now almost always based on notations in the **Unified Modeling Language (UML)**. Models help the analyst to understand the functionality of the system; they are used to communicate with customers.

Models of both new and existing system are used during requirements engineering. Models of the existing systems 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.

Models can explain the system from different 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.

The UML has many diagram types and so supports the creation of many different types of system model. Five types of UML diagrams that are the most useful for system modeling:

1. **Activity diagrams**, which show the activities involved in a process or in data processing.

2. **Use case diagrams**, which show the interactions between a system and its environment.

3. **Sequence diagrams**, which show interactions between actors and the system and between system components.

4. **Class diagrams**, which show the object classes in the system and the associations between these classes.

5. **State diagrams**, which show how the system reacts to internal and external events.

Context Model

At an early stage in the specification of a system, you should decide on the system boundaries. This involves working with system stakeholders to decide what functionality should be included in the system and what is provided by the system's environment. These decisions should be made early in the process to limit the system costs and the time needed for understanding the system requirements and design.

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. Context models normally show that the environment includes several other automated systems. However, they do not show the types of relationships between the systems in the environment and the system that is being specified.

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 since there may be pressures to develop system boundaries that increase/decrease the influence or workload of different parts of an organization.

In some cases, the boundary between a system and its environment is relatively clear. For example, where an automated system is replacing an existing manual or computerized system, the environment of the new system is usually the same as the existing system's environment.

For example, say you are developing the specification for the patient information system for mental healthcare. This system is intended to manage information about patients attending mental health clinics and the treatments that have been prescribed. In developing the specification for this system, you have to decide whether the system should focus exclusively on collecting information about consultations (using other systems to collect personal information about patients) or whether it should also collect personal patient information.

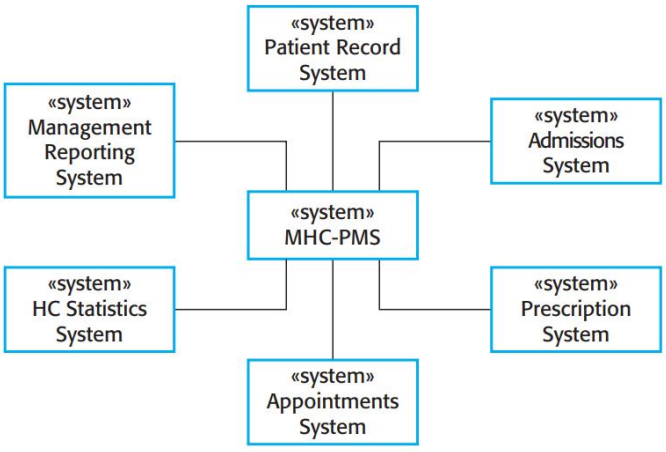


Figure 5.1 The context of the MHC-PMS

Figure 5.1 is a simple context model that shows the patient information system and the other systems in its environment. From **Figure 5.1**, you can see that the MHC-PMS is connected to an appointments system and a more general patient record system with which it shares data. The system is also connected to systems for management reporting and hospital bed allocation and a statistics system that collects information for research. Finally, it makes use of a prescription system to generate prescriptions for patients' medication.

Simple context models are used along with other models, such as business process models. These describe human and automated processes in which particular software systems are used.

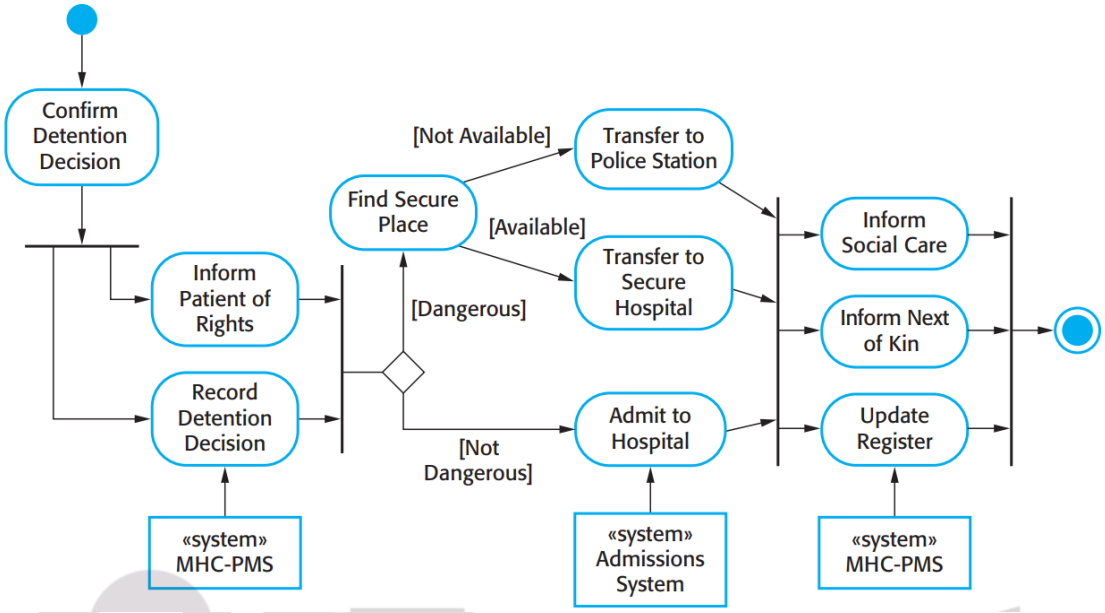


Figure 5.2 Process model of involuntary detention

Figure 5.2 is a model of an important system process that shows the processes in which the MHC-PMS is used. Sometimes, patients who are suffering from mental health problems may be a danger to others or to themselves. They may therefore have to be detained against their will in a hospital so that treatment can be administered. Such detention is subject to strict legal safeguards—for example, the decision to detain a patient must be regularly reviewed so that people are not held indefinitely without good reason. One of the functions of the MHC-PMS is to ensure that such safeguards are implemented.

Figure 5.2 is a UML activity diagram. **Activity diagrams** are intended to show the activities that make up a system process and the flow of control from one activity to another. The start of a process is indicated by a filled circle; the end by a filled circle inside another circle. Rectangles

with round corners represent activities, that is, the specific sub-processes that must be carried out.

In a UML activity diagram, arrows represent the flow of work from one activity to another. A solid bar is used to indicate activity coordination. When the flow from more than one activity leads to a solid bar then all of these activities must be complete before progress is possible.

When the flow from a solid bar leads to a number of activities, these may be executed in parallel. Therefore, in Figure 5.2, the activities to inform social care and the patient's next of kin, and to update the detention register may be concurrent.

Arrows may be annotated with guards that indicate the condition when that flow is taken. In Figure 5.2, you can see guards showing the flows for patients who are dangerous and not dangerous to society. Patients who are dangerous to society must be detained in a secure facility. However, patients who are suicidal and so are a danger to themselves may be detained in an appropriate ward in a hospital.

Interaction models

All systems involve interaction of some kind. This can be user interaction, which involves user inputs and outputs, interaction between the system being developed and other systems or interaction between the components of the system.

Types of interactions that can be represented in a model:

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.

There are two related approaches to interaction modeling:

1. **Use case modeling**, which is mostly used to model interactions between a system and external actors (users or other systems).

2. **Sequence diagrams**, which are used to model interactions between system components, although external agents may also be included.

Use case models and sequence diagrams present interaction at different levels of detail and so may be used together.

Use case modeling

A use case can be taken as a simple scenario that describes what a user expects from a system. Each use case represents a discrete task that involves external interaction with a system. In its simplest form, a use case is shown as an ellipse with the actors involved in the use case represented as stick figures.

Figure 5.3 shows a use case from the MHC-PMS that represents the task of uploading data from the MHC-PMS to a more general patient record system. This more general system maintains summary data about a patient rather than the data about each consultation, which is recorded in the MHC-PMS.

Figure 5.3 Transfer-data use case

There are two actors in this use case: the operator who is transferring the data and the patient record system. The stick figure notation was originally developed to cover human interaction but it is also now used to represent other external systems and hardware. Formally, use case diagrams should use lines without arrows as arrows in the UML indicate the direction of flow of messages. Obviously, in a use case message pass in both directions.

Use case description in a tabular format:

Use case title	**Transfer data**
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.
Actor(s)	Medical receptionist, patient records system (PRS)
Preconditions	Patient data has been collected (personal information, treatment summary); The receptionist must have appropriate security permissions to access the patient information and the PRS.

Postconditions	PRS has been updated
Main success scenario	1. Receptionist selects the "Transfer data" option from the menu. 2. PRS verifies the security credentials of the receptionist. 3. Data is transferred. 4. PRS has been updated.
Extensions	2a. The receptionist does not have the necessary security credentials. 2a.1. An error message is displayed. 2a.2. The receptionist backs out of the use case.

Sequence diagrams

Sequence diagrams in the UML are primarily used to model the interactions between the actors and the objects in a system and the interactions between the objects themselves.

A sequence diagram shows the sequence of interactions that take place during a particular use case or use case instance. Figure 5.6 is an example of a sequence diagram that illustrates the basics of the notation. This diagram models the interactions involved in the View patient information use case, where a medical receptionist can see some patient information.

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. The rectangle on the dotted lines indicates the lifeline of the object concerned (i.e., the time that object instance is involved in the computation). We read the sequence of interactions from top to bottom. The annotations on the arrows indicate the calls to the objects, their parameters, and the return values.

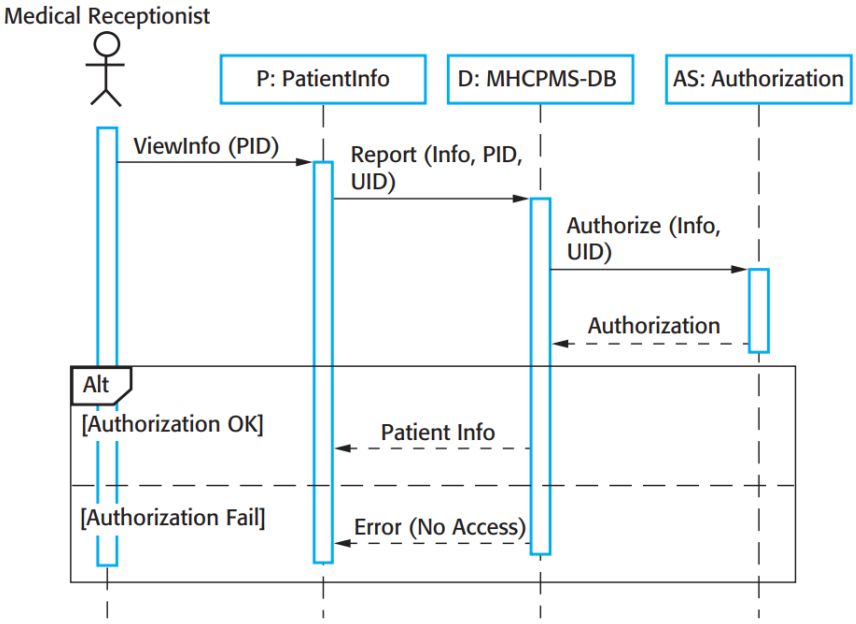


Figure 5.6 Sequence diagram for View patient information

Figure 5.6 represents Sequence diagram as follows:

1. The medical receptionist triggers the ViewInfo method in an instance P of the PatientInfo object class, supplying the patient's identifier, PID. P is a user interface object, which is displayed as a form showing patient information.

2. The instance P calls the database to return the information required, supplying the receptionist's identifier to allow security checking (at this stage, we do not care where this UID comes from).

3. The database checks with an authorization system that the user is authorized for this action.

4. If authorized, the patient information is returned and a form on the user's screen is filled in. If authorization fails, then an error message is returned.

Behavioral models

Behavioral models are models of the dynamic behavior of the 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. Stimuli is of two types:

1. Data Some data arrives that has to be processed by the system.

2. Events Some event happens that triggers system processing. Events may have associated data but this is not always the case.

Data-flow diagrams

Data-flow diagrams (DFDs) are system models that show a functional perspective where each transformation represents a single function or process. DFDs are used to show how data flows through a sequence of processing steps. For example, a processing step could be the filtering of duplicate records in a customer database. The data is transformed at each step before moving on to the next stage. These processing steps or transformations represent software processes or functions where data-flow diagrams are used to document a software design.

Data-flow models are useful because tracking and documenting how the data associated with a particular process moves through the system helps analysts and designers understand what is going on. Data-flow diagrams are simple and intuitive and it is usually possible to explain them to potential system users who can then participate in validating the model.

Event-driven modeling

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. Event-driven models can be created using UML state diagrams:

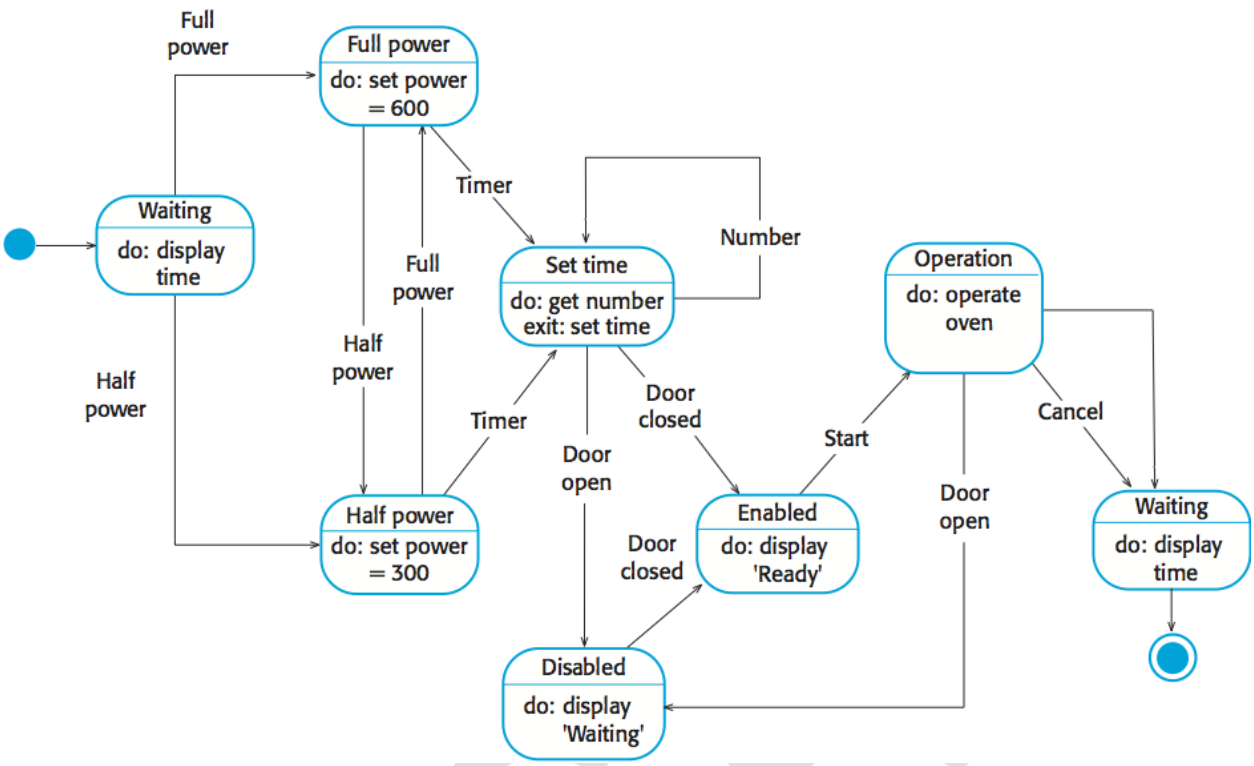


Figure 5.16 State diagram of a microwave oven

In UML state diagrams, rounded rectangles represent system states. They may include a brief description (following 'do') of the actions taken in that state. The labeled arrows represent stimuli that force a transition from one state to another. You can indicate start and end states using filled circles, as in activity diagrams.

From **Figure 5.16**, you can see that the system starts in a waiting state and responds initially to either the full-power or the half-power button. Users can change their mind after selecting one of these and press the other button. The time is set and, if the door is closed, the Start button is enabled. Pushing this button starts the oven operation and cooking takes place for the specified time. This is the end of the cooking cycle and the system returns to the waiting state.

Structured Model

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

UML **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.

Class diagrams in the UML can be expressed at different levels of detail. When you are developing a model, the first stage is usually to look at the world, identify the essential objects, and represent these as classes. The simplest way of writing these is to write the class name in a box. You can also simply note the existence of an association by drawing a line between classes. For example, **Figure 5.8** is a simple class diagram showing two classes: Patient and Patient Record with an association between them.

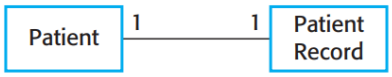


Figure 5.8 UML classes and association

In this example, each end of the association is annotated with a 1, meaning that there is a 1:1 relationship between objects of these classes. That is, each patient has exactly one record and each record maintains information about exactly one patient.

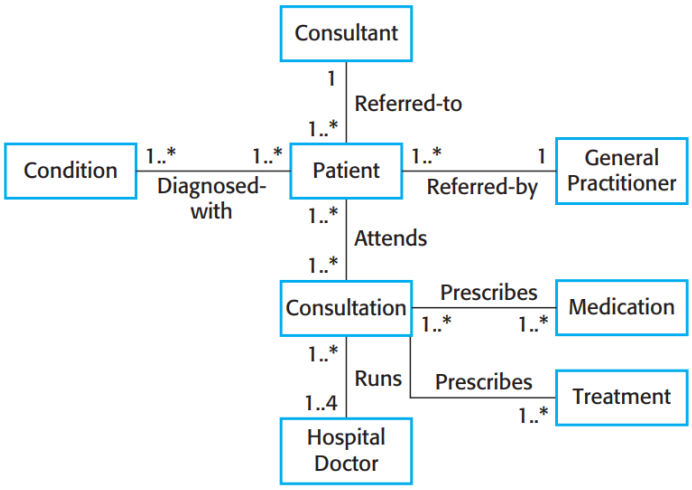


Figure 5.9 Classes and associations in the MHC-PMS

Figure 5.9 develops this type of class diagram to show that objects of class Patient are also involved in relationships with a number of other classes.

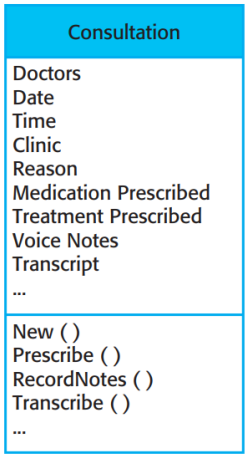
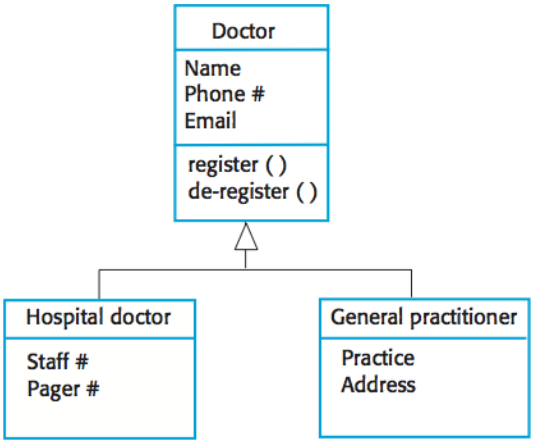


Figure 5.10 The consultation class

When showing the associations between classes, it is convenient to represent these classes in the simplest possible way. To define them in more detail, you add information about their attributes (the characteristics of an object) and operations (the things that you can request from an object). For example, a Patient object will have the attribute Address and you may include an operation called ChangeAddress, which is called when a patient indicates that they have moved from one address to another. In the UML, you show attributes and operations by extending the simple rectangle that represents a class. **This is illustrated in Figure 5.10 where**:

1. The name of the object class is in the top section.
2. The class attributes are in the middle section. This must include the attribute names and, optionally, their types.
3. The operations (called methods in Java and other OO programming languages) associated with the object class are in the lower section of the rectangle.

Generalization is an everyday technique that we use to manage complexity. In modeling systems, it is often useful to examine the classes in a system to see if there is scope for generalization. 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

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.

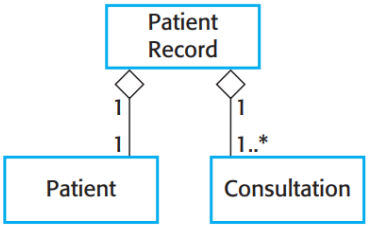


Figure 5.13 The aggregation association

Data Model

Most large software systems make use of a large database of information. In some cases, this database is independent of the software system. In others, it is created for the system being developed. An important point of system modeling is defining the logical form of data processed by the system. The most widely used data modeling technique is Entity-Relation-Attribute modeling (ERA modeling), which shows the data entities, their associated attributes and the relation between these entities. Entity – relationship models have been used widely in database design.

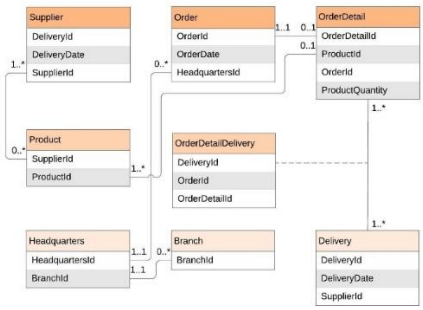


Figure – Data model of Online Shopping