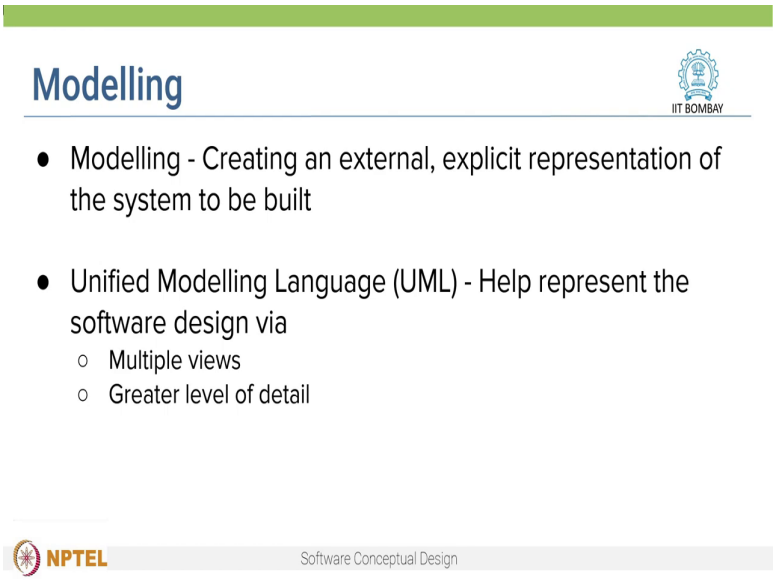
Software Conceptual Design

Dr. Sridhar Iyer
Dr. Prajish Prasad
Dr. T. G. Lakshmi
Department of Computer Science and Engineering
Indian Institute of Technology, Bombay

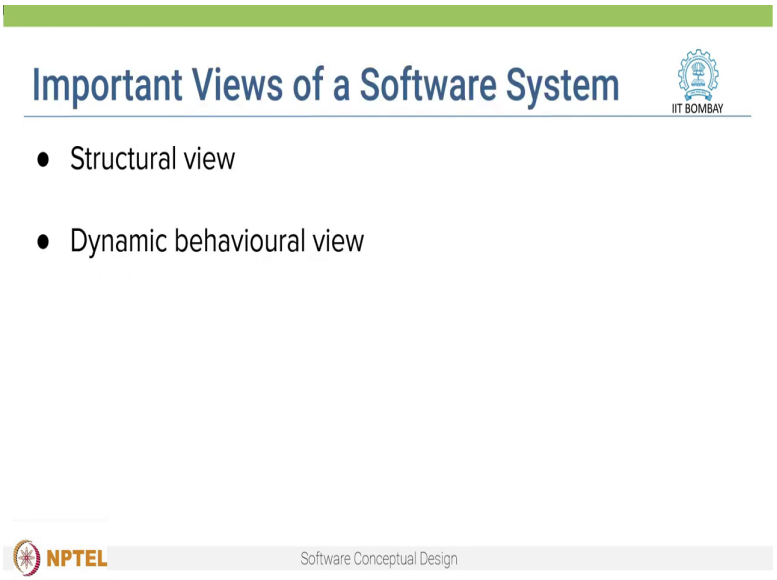
Lecture - 16
Unified Modelling Language (UML)

(Refer Slide Time: 00:05)



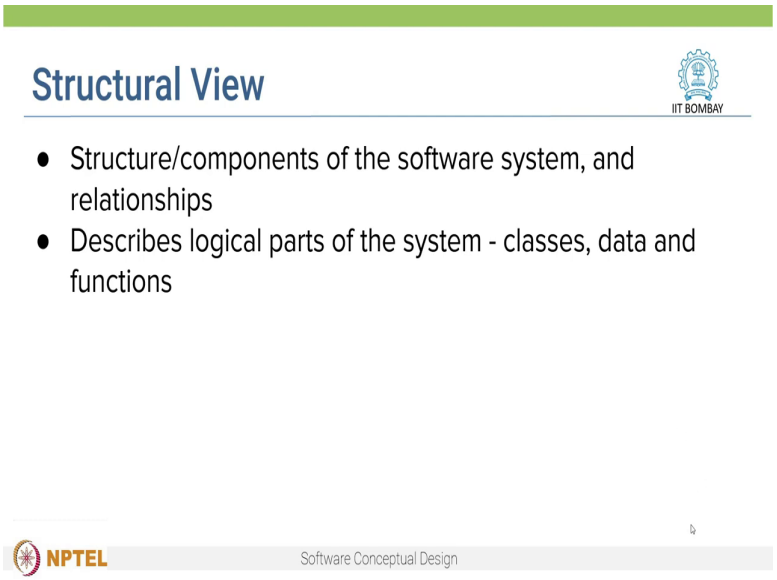
In the previous video, we talked about modelling a software design. Modelling is a way of creating an external explicit representation of the system to be built. In this video, we will look at the Unified Modelling Language or the UML diagrams. These diagrams can help us represent the software design via multiple views and at a greater level of detail.

(Refer Slide Time: 00:41)



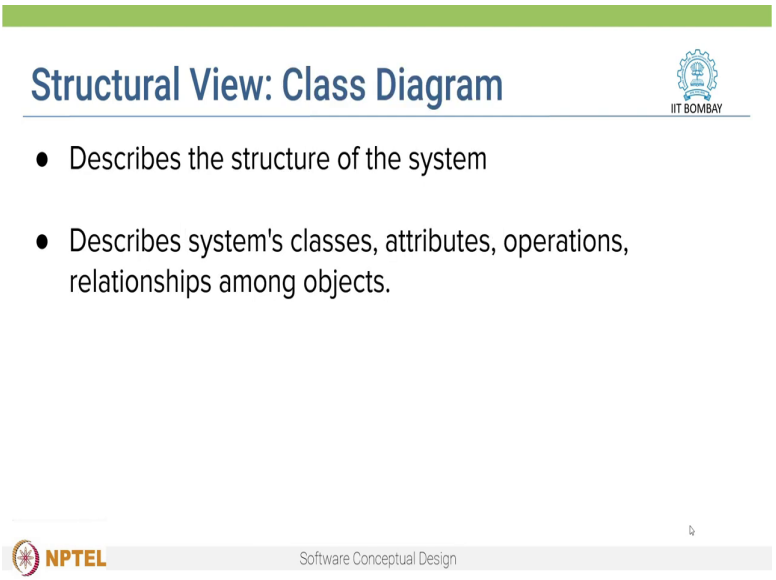
Let us look at what these views are. Two important views of a software system are the Structural view and the Dynamic behavioural view.

(Refer Slide Time: 00:52)



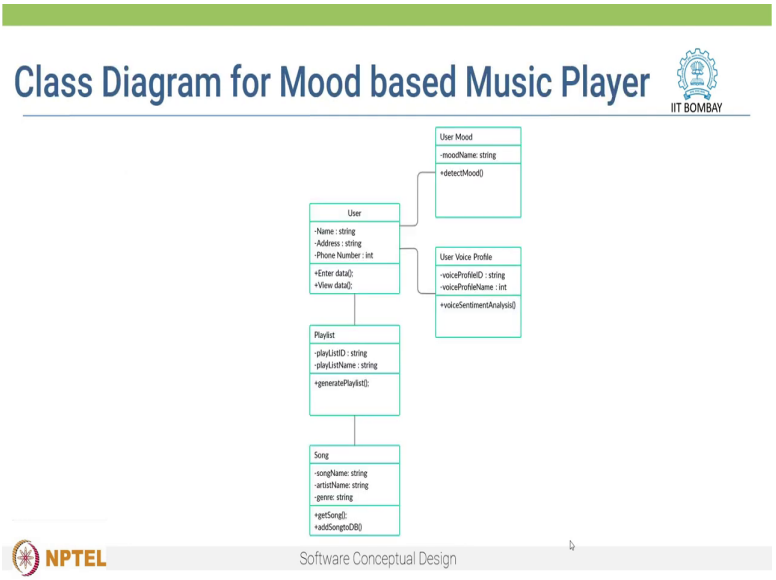
So, what is the structural view? The structural view defines the structure of the software system. They describe the things in the system and their relationships to other things. The structural view describes the logical parts of the system, including the classes and relevant data and functions in these classes. They do not describe the time-dependent behaviour of the system.

(Refer Slide Time: 01:26)



One such diagram which represents the structural view is the Class Diagram. The class diagram is a type of a static structure diagram that describes the structure of a system by showing the systems classes, the attributes, the operations or methods and the relationship among these objects.

(Refer Slide Time: 01:53)

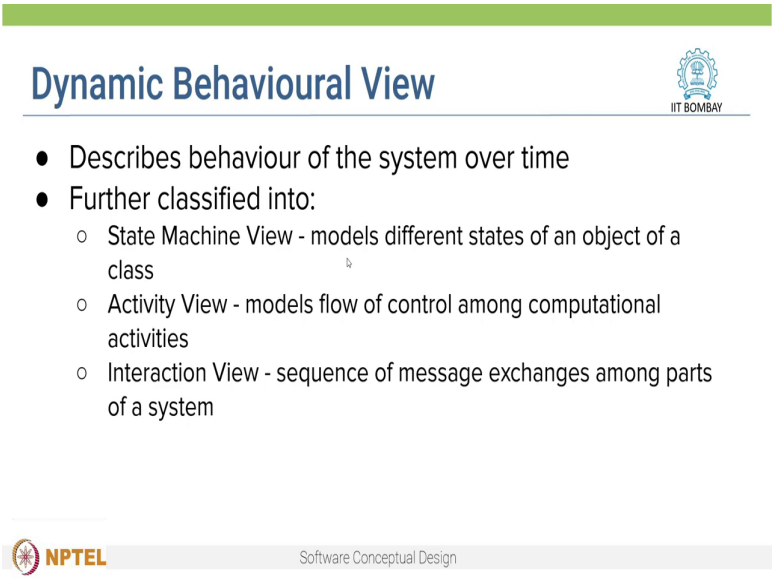


The diagram shown on this slide describes the Class Diagram of the mood based Music Player. The class diagram is the main building block of object-oriented modelling. In class diagrams, the classes are presented with boxes that contain 3 compartments. The top

compartment contains the name of the class, the middle compartment contains the attributes of the class, and the bottom compartment contains the operations of the class, which can be executed.

We see that the class diagram models relevant classes like the User, the User Voice Profile, the Playlists, the Songs and so on. Each of these classes contain data members and functions necessary to realise certain behaviours of the mood-based music player. For example, the user mood class has a data attribute moodName, which stores the current mood of the user. The detectMood function, in the user mood class updates the moodName variable based on the User and the User Voice Profile Data.

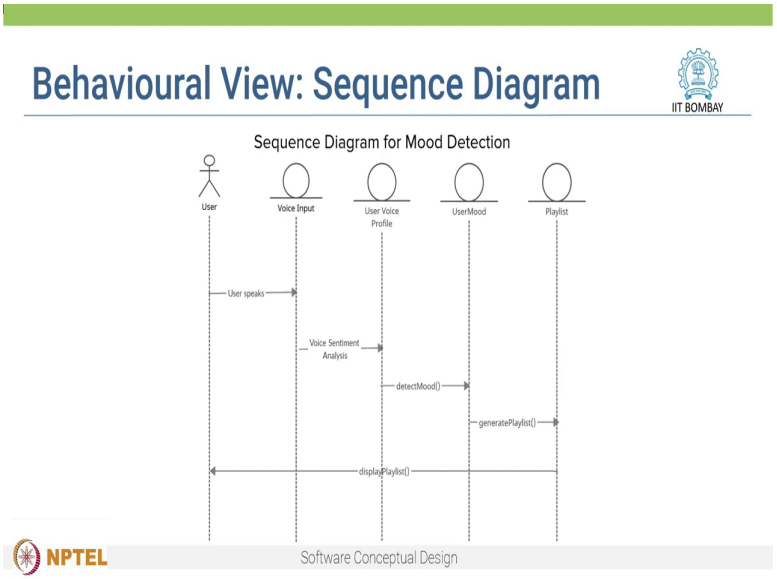
(Refer Slide Time: 03:29)



The Dynamic Behaviour View describes the behaviour of the system over time. Various views include the state machine view, the activity view and the interaction view. The state machine view models the possible life histories of an object of a class. The activity view shows the flow of control among computational activities involved in performing a calculation or a workflow.

The interaction view describes sequence of messages exchanged among different parts of a system. It gives a holistic view of the behaviour in a system by showing the flow of control across many objects.

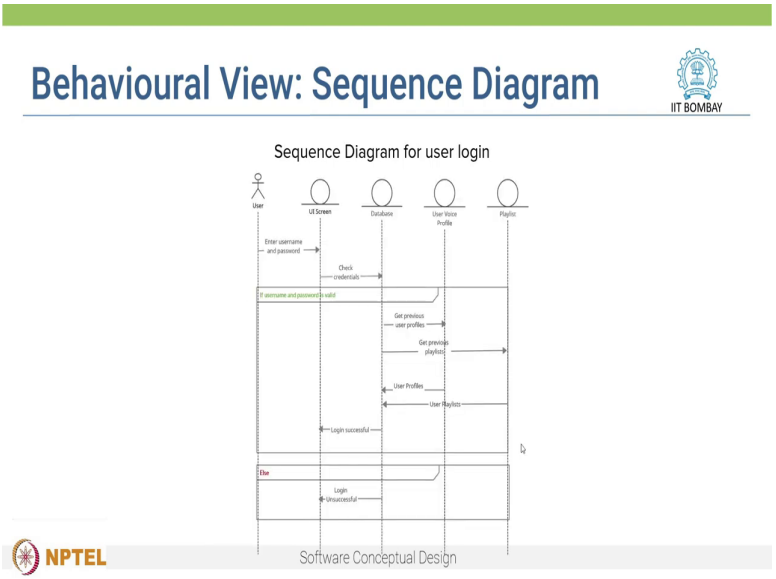
(Refer Slide Time: 04:28)



The behaviours corresponding to the requirements of a system can be implemented using Sequence Diagrams. The diagram on this slide represents the sequence diagram which describes the behaviour of the mood detection requirement for the mood based music player system. The classes from the class diagram are represented in the sequence diagram on the top. And they have parallel vertical lines called the lifelines.

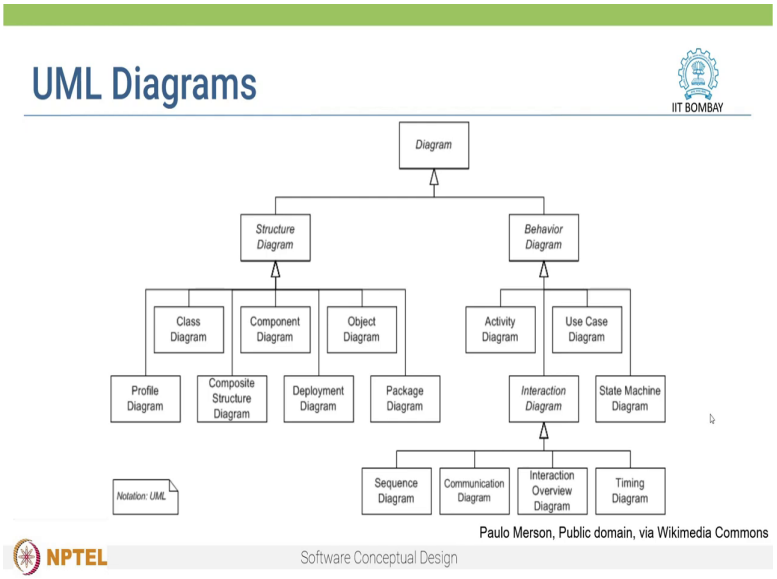
The horizontal arrows which you see are the messages exchanged between them in the order that they occur. These messages can be functions from the class diagram, such as detectMood, generatePlaylist and so on. This sequence of messages correspond to the implementation of a given requirement or sub-requirement. In a similar fashion, sequence diagrams can be constructed to model other requirements as well.

(Refer Slide Time: 05:52)



For example, the sequence diagram in this slide models the login feature for the mood based music player.

(Refer Slide Time: 06:12)



There are several other UML diagrams, such as state charts, communication diagrams, activity diagrams, etcetera which can be used to adequately model a given software design. Details of various UML diagrams can be found in the extra resources provided in this week. During the software design process, a subset of these UML diagrams are used to create the design model.