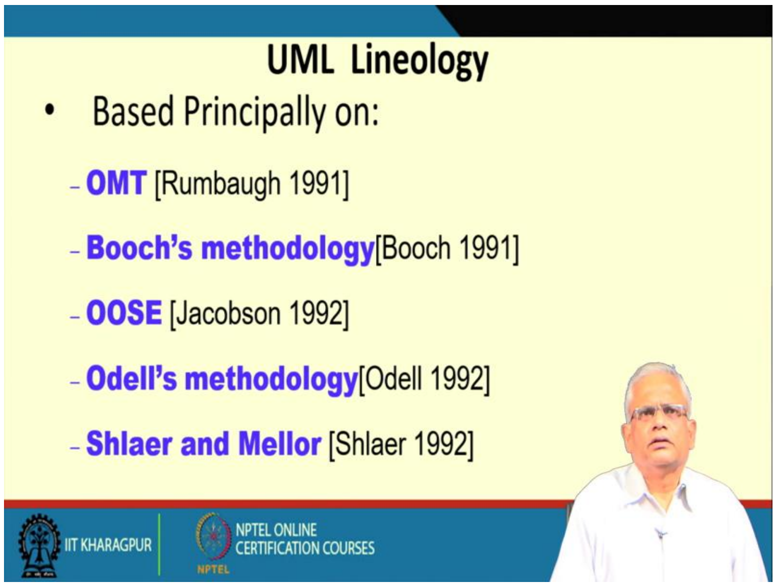
Object – Oriented System Development using UML, Java and Pattern
Professor. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Basic Concepts in UML
Lecture 02

(Refer Slide Time: 00:14)



UML Lineology

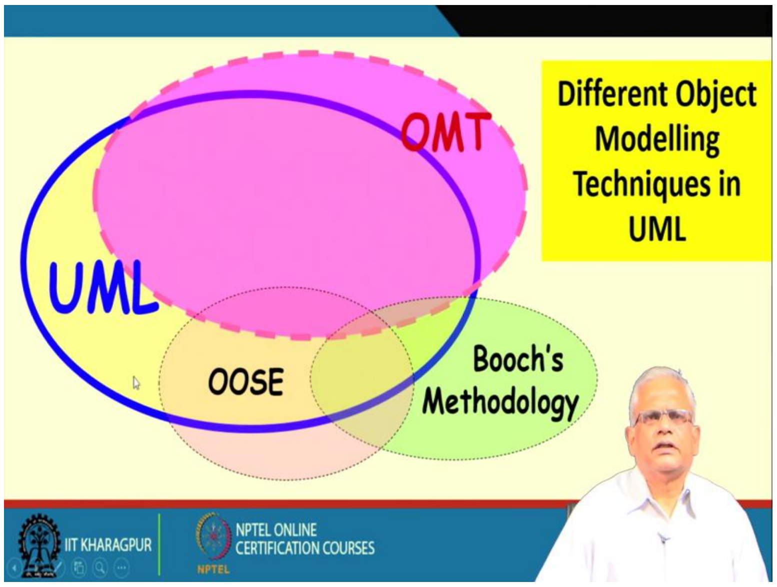
- Based Principally on:
  - **OMT** [Rumbaugh 1991]
  - **Booch's methodology**[Booch 1991]
  - **OOSE** [Jacobson 1992]
  - **Odell's methodology**[Odell 1992]
  - **Shlaer and Mellor** [Shlaer 1992]

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL

Welcome to this lecture. In the last lecture, we had discussed some very basic aspects. We had said that, in this course we are going to discuss UML first. The unified modeling language is a set of notations and grammar rules just like any language. UML is a language just like any language but then it is a graphical language. In English, you use alphabets and grammar rules, but here, we will use graphical symbols and then some rules about how to combine the symbols to come up with some meaningful models of object oriented systems. In the last lecture, we had discussed about how UML came about we said that a large number of design methodologies existed in the early 90s and these were prominent techniques, in that sense, in the sense that they were used extensively.

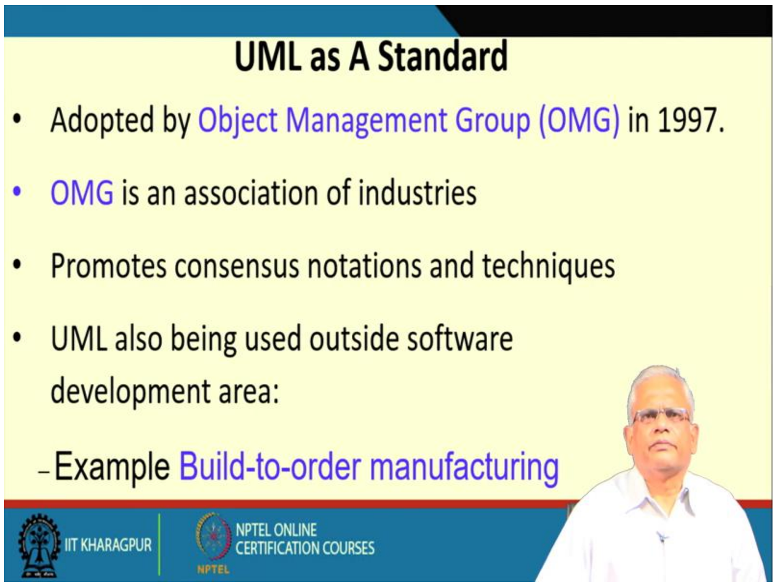
Many projects were being developed using these techniques. Some of the prominent techniques are OMT, the object modeling technique, the Booch's methodology. OMT was proposed by Rumbaugh Blaha etcetera and Booch's methodology was proposed by Grady Booch. Object oriented software engineering, this is another design methodology proposed by Jacobson. Odell's methodology by Odell. Shlaer and Mellor methodology by Shlaer and Mellor 92 and each of these had different symbols and rules of combining the symbols to get meaningful models and UML drew upon these various methodologies that existed. Many of these symbols were assimilated into UML and of course UML propose some symbols which are not there in any of this methodologies, symbols, grammar rules and models which are not there.

(Refer Slide Time: 02:57)


Different Object Modelling Techniques in UML
OMT
UML
OOSE
Booch's Methodology
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

If we pictorially show, how the different models were used in UML. As you can see in this diagram UML to a large extent drew upon the OMT. OMT was one of the most popular technique that was used in 90s and large part of UML is similar to OMT. If you know OMT, you can find the similarity in UML. But, there are some parts which are not there in UML but have been adapted from object oriented software engineering, Jacobson’s method and the Booch’s methodology and also you can see here, that some part of UML is not present in any of the methodology, these are some new notations and models which have been put into UML.

(Refer Slide Time: 03:57)


UML as A Standard

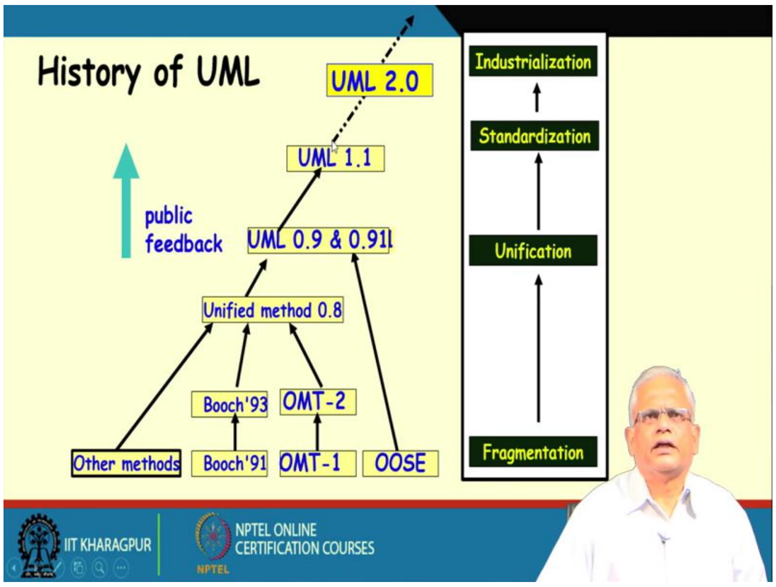
- Adopted by Object Management Group (OMG) in 1997.
- OMG is an association of industries
- Promotes consensus notations and techniques
- UML also being used outside software development area:
  - Example Build-to-order manufacturing

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL

It was adopted, by the object management group in 1997. OMG is association of industries and they don’t really formulate standards, but then they popularize certain techniques and the member companies. In these association they start using it and it becomes a de facto standard, because the use becomes widespread and the UML was adopted by OMG. Since 1997, UML has become very popular year after year and now it is almost universally used all over in academics and industry and not only it is used in software modeling, software development, but also the notations are so elegant and popular that even in other industries, the UML is used.

For example, in many industry we have this build to order manufacturing, where the customer places the order for some equipment maybe a car, maybe a machinery and the company builds according to the customer specific order and the way, the car or a machine is described is often based on UML, because this has turned out to be a very elegant modeling language. It becomes not only easy for the customers to learn it and specify what they really want, but also it provides a very elegant model which the manufacturer can use to give a car or a machine that complies to the customer’s requirement.

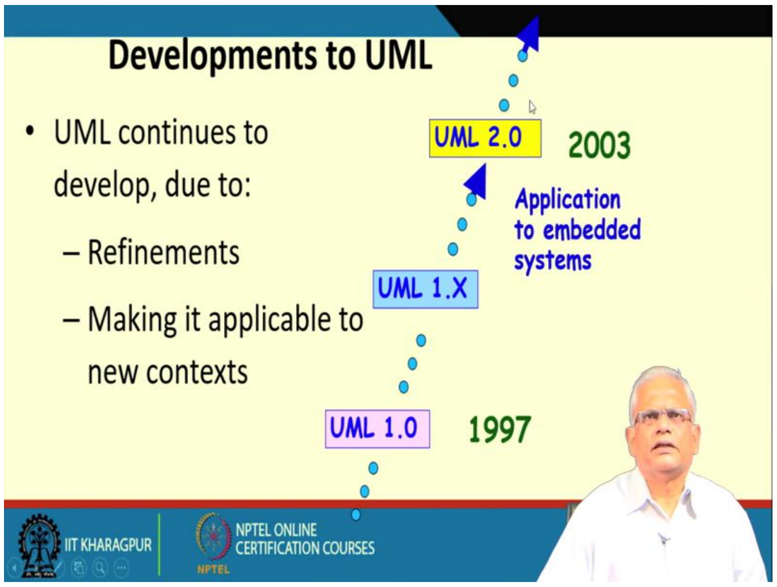
(Refer Slide Time: 06:23)



As we were discussing, in the early 90s many techniques were existing. Not only they were existing, there were also several versions each of these techniques were evolving, they are bringing out new notions etc. and around 1997, the UML 0.8 was proposed and UML 1 was accepted by OMG 1997 and since then it has still been growing, it has been evolving. So, initially there was a fragmented set of notions, different companies were using different notations and even in the same company different notations were used and then unification, a set of notations which are standard, all companies, all projects use the same notation and this has helped, reusing design getting adjusted to a new design environment or if you learn a design in your college, you go to the industry and find that the same one is being used.

So, that is the advantage here, reuse and reuse of design solutions and you can straight away use the technique once you have learnt it. But then it has been receiving feedback on how to improve, what is lacking to be able to design some specific systems and based on that UML has been evolving and one major release of UML is UML 2.0. It has been evolving with different versions, incrementally. But a major release in UML 2.0, because of the many industry problems specifically the embedded systems required different sets of notations, for events, signals and so on and UML 2.0 was brought out.

(Refer Slide Time: 08:55)



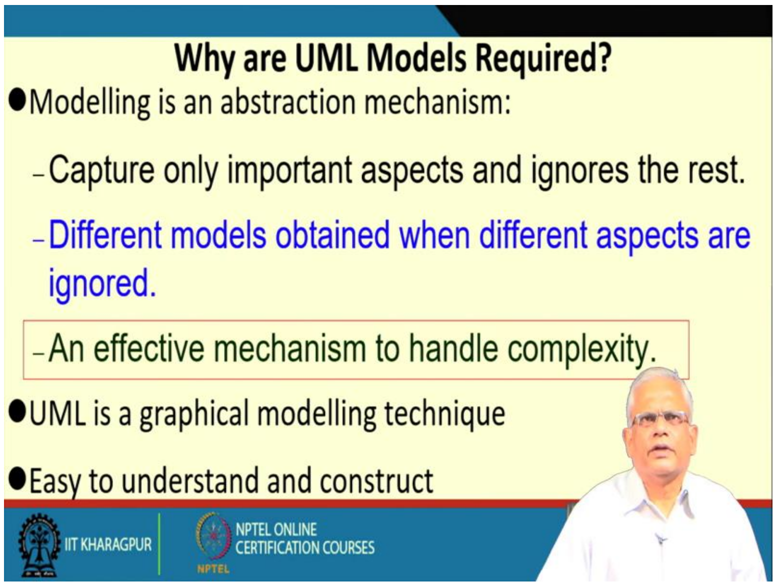
Developments to UML

- UML continues to develop, due to:
  - Refinements
  - Making it applicable to new contexts

UML 2.0
2003
Application to embedded systems
UML 1.X
UML 1.0
1997
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

UML still continues to develop starting with UML 1.0. In 1997, new versions have been coming up. These are refinements to the UML and a major revision to UML in 2003, which is UML 2.0 and the main improvements here to UML are application to embedded systems, the embedded systems have concurrency, events, signals, reaction to events and so on and these have been new notations have been incorporated. Some of the notations have been refined and UML 2.0 is right now being used extensively and as part of this course we will start with simple UML 1.X notations and then we will see what are the new things that have happened in UML 2.0 and it still continues to grow.

(Refer Slide Time: 10:06)



Why are UML Models Required?

- Modelling is an abstraction mechanism:
  - Capture only important aspects and ignores the rest.
  - Different models obtained when different aspects are ignored.
  - An effective mechanism to handle complexity.
- UML is a graphical modelling technique
- Easy to understand and construct

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

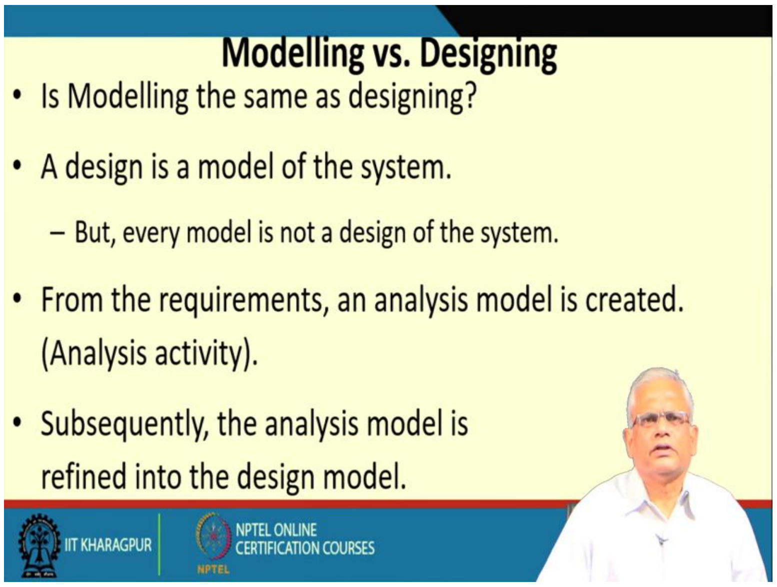
Before we learn the nitty gritty of UML, let us be clear about one issue that why do we need UML models, how does it help if we model something. The answer to that question is that modeling is an abstraction mechanism. An abstraction mechanism is one, where we look at a problem and then we create a simple model of the problem, the description of a problem. A non-trivial problem maybe hundreds of pages long with various details. It becomes very difficult for somebody to read those hundred pages and have a meaningful understanding of what the system is required to do. In this situation, modeling comes in handy.

In different models, we capture the important aspects from a hundred page document we distill out a few graphical models, which actually capture the essence of the system that is required, you just look at the model and you have a very good understanding, what is required and if you read those hundred pages you get lost in the details, you do not get to know what exactly is required and if you do not know what exactly is required, how do you design and we do not create one model, we create different models that capture different aspects. Focus on one aspect, ignore the rest of the aspect and we model that aspect only and this is the way we handle complexity by creating model.

A hundred-page document is a description of a complex system because this describes the system entirely. We create a model that discusses, that captures some issue, some specific aspect of the system and by looking at the model, we become absolutely clear what is required. Just to think of a building, if a contractor is given the task of coming up with a large building, one way is that we describe him about the building, a hundred storey building and what are the things that happened in different floors, what will be the, how does it look like internally and so on. But then, the contractor does not get a full idea, if we give him a hundred-page document describing the building. We need to have a model of the building worked out, that is we give the layout, what is the frontal design, what is the floor plan and so on.

So, we give him a set of models of the building and based on that, he can construct the building. The same thing here with program, just a verbal description of the system, it is very difficult to come up with a good solution, we need to have modeling from the problem description and this modeling can be used by the programmer to come up with the exact solution that is required. A model in that sense we can say that it is a simplified representation of the actual thing and it is an effective mechanism to handle complexity. UML is a graphical modeling technique, we will be given a text description of a system, we will create a graphical model. It will be easy to understand the model and to construct the solution based on the UML model.

(Refer Slide Time: 15:19)



Modelling vs. Designing

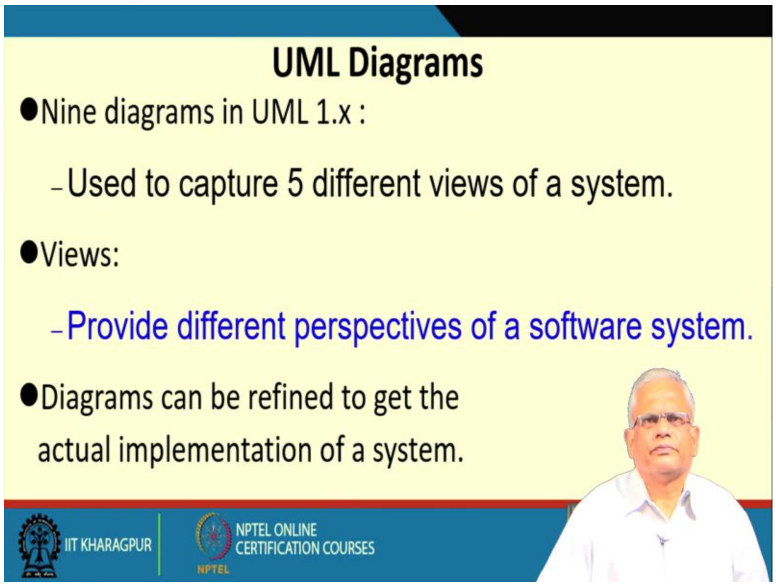
- Is Modelling the same as designing?
- A design is a model of the system.
  - But, every model is not a design of the system.
- From the requirements, an analysis model is created. (Analysis activity).
- Subsequently, the analysis model is refined into the design model.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL

Another very basic thing that we must be clear at this point in the course, is that what is the difference between modelling and designing, are they the same thing? Is modeling the same thing as designing and designing the same thing as modeling? No, not really. A design is certainly a model of the system. A design is a simplified representation of the actual code and the code is written according to the design. But every model is not the design, we have analysis model. Every model is not the design of the system, initially we start with an analysis model through analysis activity, which is not a design model. The analysis model cannot be directly translated to code or code cannot be easily written from the analysis model. But we transform the analysis model into a design model.

So, a design is one type of model and model can be either design model or an analysis model. An analysis model is just a model of the problem. Whereas a design model is the model of the solution. Given the design model we can straight away have the program written based on the design. But given the analysis model, we cannot really write the code. So, we need to transform the model of the problem domain, that is the analysis model into the design model and as part of this course, the design process will first create the analysis model and then we will transform that into the design model and that will be our design methodology, which we will discuss after discussing about UML.

(Refer Slide Time: 17:46)



UML Diagrams

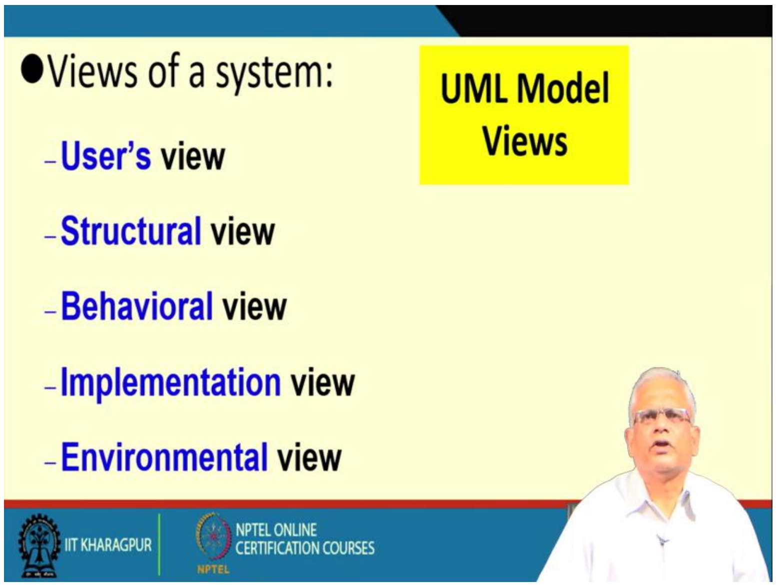
- Nine diagrams in UML 1.x :
  - Used to capture 5 different views of a system.
- Views:
  - Provide different perspectives of a software system.
- Diagrams can be refined to get the actual implementation of a system.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL

In UML 1.x, nine diagram exist as I was saying that UML 2.0 was a major revision to UML 1.x and there were four more new diagrams were added, 13 diagrams and initially there are nine diagrams, most of these diagrams have been carried on into the UML 2.x. But then there are small changes in these diagrams, we use these nine diagrams to capture five different views of a system. What is a view of a system? A view gives a perspective of the software system. A perspective is basically for what purpose we use the model. For example, we might have a user's view.

The user's view is, how does the system understood by the common user. The common user understands the system in the form of its functionalities, the external behavior of the system, what he can do with it, can he create a log in, can he carry out some operations? So, those are the user's view. What are the operations you can carry using the system. Similarly, we have the internal views like a structural view, that how the system is structured and so on. So, there are five different views of the system, we will shortly look at what are the views of the system and we have different diagrams in UML to model these different views of the system.

(Refer Slide Time: 19:51)



- Views of a system:
  - User's view
  - Structural view
  - Behavioral view
  - Implementation view
  - Environmental view

UML Model Views

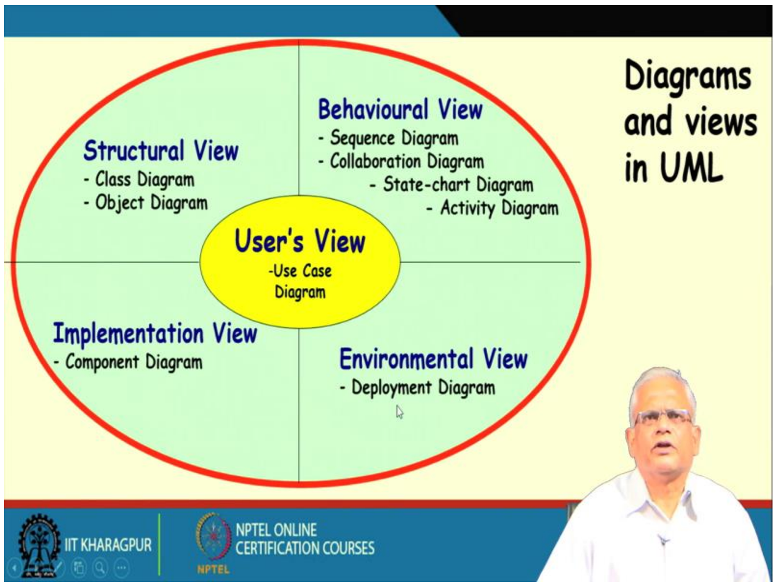
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL

The five different views are the user' s view, this is the most basic view, because after all the user describes the problem and we need to captures the user's view, because that is the basic thing that the user needs and this view once we create the model of the user's view, the user should be able to understand it and then say that whether that actually fits into what he was saying. So, the user's view is very important and invariably this is the first model that is created, the user's view. Because we first create the model based on the user's view get it ratified by the user or if the user says that some changes are needed, we incorporate the changes in the user's view until the user agrees.

So, this is one of the very fundamental views of a system, first to be constructed in any development process is the user's view. The structural view, is that what is the structure of the system, what are the classes, how are they packaged and so on. The third one is a behavioral view, here given an input to the system or let's say the user invokes a functionality. How do the classes interact among themselves? How do the different object that exist, they interchange messages and then come up with the required behavior? So, that is the behavioral view and even in the behavioral view we have, how does the states of different objects changes as they receive different inputs?

So, those are the behavioral views, we have a set of diagrams to model the behavioral view and then we have the implementation view. In the implementation view, we have how is the implementation structured? Are there packages to be deployed, what is the configuration on those packages and so on. And the environmental view is, what are the different physical components of the system. Are there number of processors? Are there I/O units and how are the different parts of the solution, the program code going to deployed in these different components. So, that is the environmental view.

(Refer Slide Time: 22:52)



Diagrams and views in UML

Structural View
- Class Diagram
- Object Diagram

Behavioural View
- Sequence Diagram
- Collaboration Diagram
- State-chart Diagram
- Activity Diagram

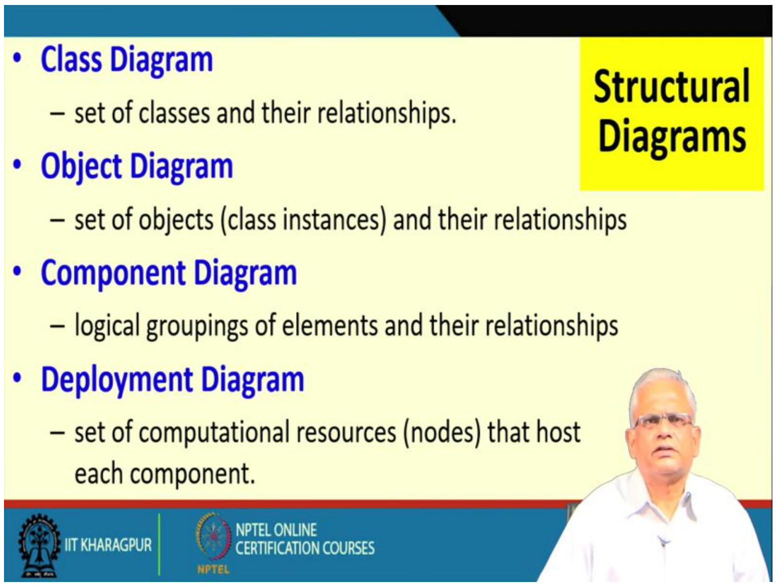
User's View
-Use Case Diagram

Implementation View
- Component Diagram

Environmental View
- Deployment Diagram

This diagram, shows the different views, the five different views. The central view is the user’s view. This is the first view constructed and the other views are constructed based on the user’s view and the diagram that is used to model the user's view is the use case diagram and this diagram we will discuss first in this lecture. We have the structural view modeled using class and object diagrams. The behavioral view, sequence diagram, collaboration diagram, state chart diagram, activity diagram these are used to model the behavioral view. The implementation view, in terms of component diagrams, what are the different executable components and how are these components deployed on different physical computing elements like different processors I/O nodes and so on, that is a deployment diagram.

(Refer Slide Time: 24:07)



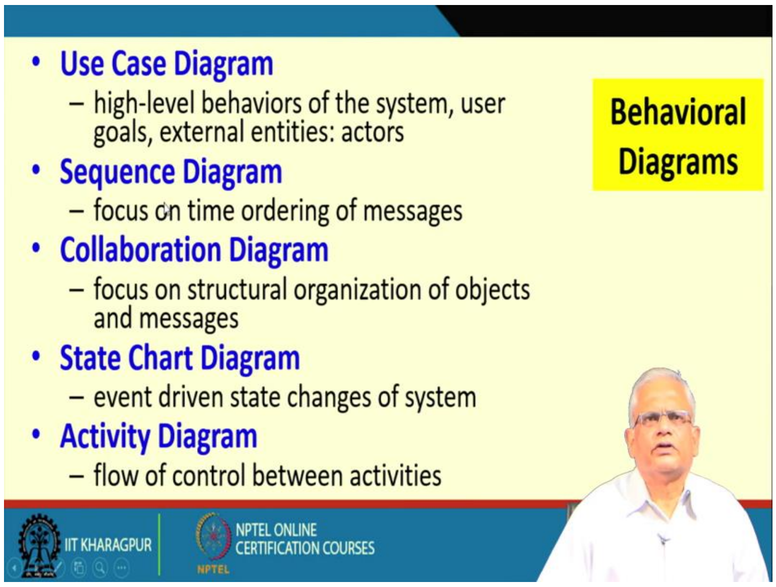
- **Class Diagram**
  - set of classes and their relationships.
- **Object Diagram**
  - set of objects (class instances) and their relationships
- **Component Diagram**
  - logical groupings of elements and their relationships
- **Deployment Diagram**
  - set of computational resources (nodes) that host each component.

Structural Diagrams

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

As you could see that there are many diagrams here. Nine diagrams in the UML 1.x and to simplify our understanding, we will first start with UML 1.x diagrams and till what are the changes that have been made in 2.x, class diagram we capture, what are the classes and their relations. In object diagram, we capture what are the objects and how do they interact, what are the relations between them. In component diagram, what are the different elements that are grouped, the deployment diagram? What are the nodes that host the components?

(Refer Slide Time: 25:02)



- **Use Case Diagram**
  - high-level behaviors of the system, user goals, external entities: actors
- **Sequence Diagram**
  - focus on time ordering of messages
- **Collaboration Diagram**
  - focus on structural organization of objects and messages
- **State Chart Diagram**
  - event driven state changes of system
- **Activity Diagram**
  - flow of control between activities

Behavioral Diagrams

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

The use case diagram is the behavioral diagrams, here is the user's view and the sequence diagram how are the messages ordered. Collaboration diagram is focus under structural organization. State chart diagram, how the system state changes and the activity diagram how the flow of control occurs different activities. We will discuss all these diagrams as part of this course, that is the first thing we will do before looking at the design process and coming in up with design solutions to a set of problems and once we do that these different diagrams will fall into place that what exactly are the capabilities of this diagrams. What do they really model? What are the notations used and so on?

We will start looking at those aspects in the next lecture. With this brief introduction to UML the different views, the five different views and the different diagrams to model this views. We will stop here and we will start with looking at the use case diagram to model the user's view of the system because after all, that is the central view and every design process starts with the user’s view of the system. We will stop now, thank you.