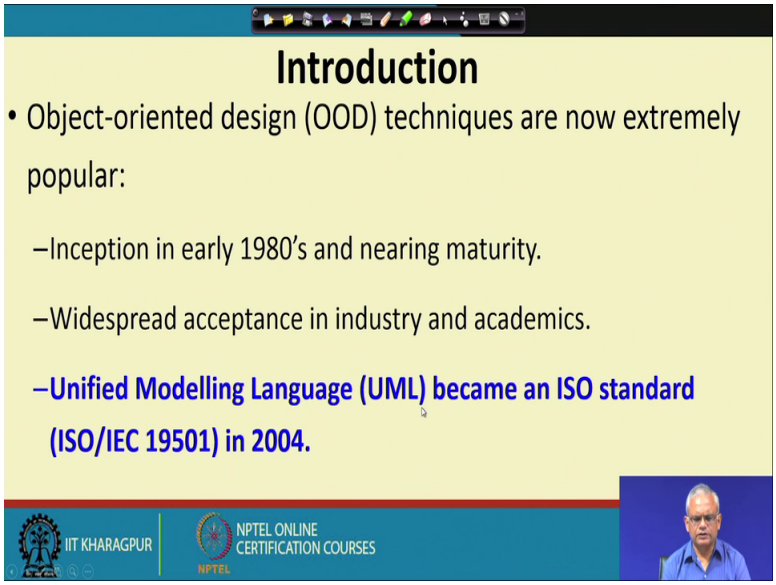
Software Engineering
Prof. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 30
Structured Design Examples

Let us see how to carry out the Object Oriented Design. We have discussed about the procedural design. And, now we look at the netegrity of object oriented design. For, object oriented design we use UML as the notation.

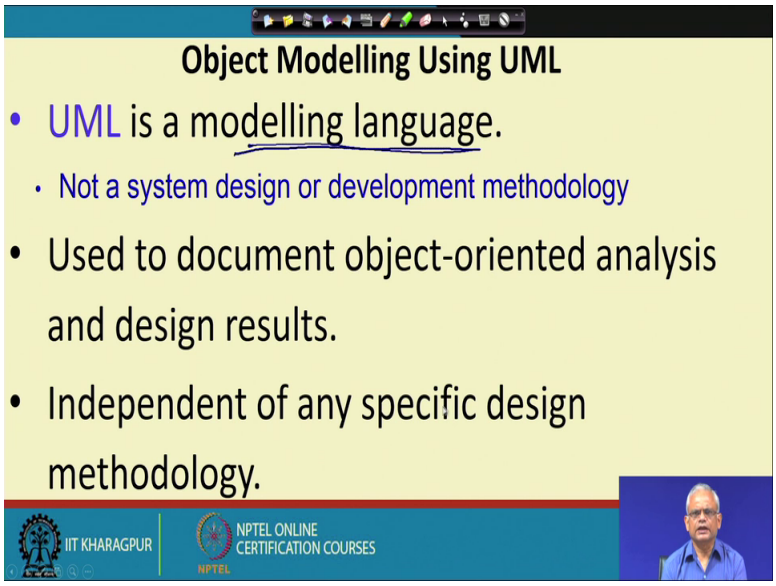
(Refer Slide Time: 00:43)



Let us get started with that nowadays, object-oriented design techniques have become extremely popular, used extensively the initial work in object orientation is done in 1980s.

And, now nearing maturity there is widespread acceptance in industry and academics and UML is the modelling language using which design is done, stands for unified modelling language and that has become an ISO standard in 2004. And therefore, the object oriented design is much more standardized than the procedural design. Now, let us see how to go about developing the UML model? Why do we develop a model UML model? And, then how do we get the design finally, from the model?

(Refer Slide Time: 01:49)

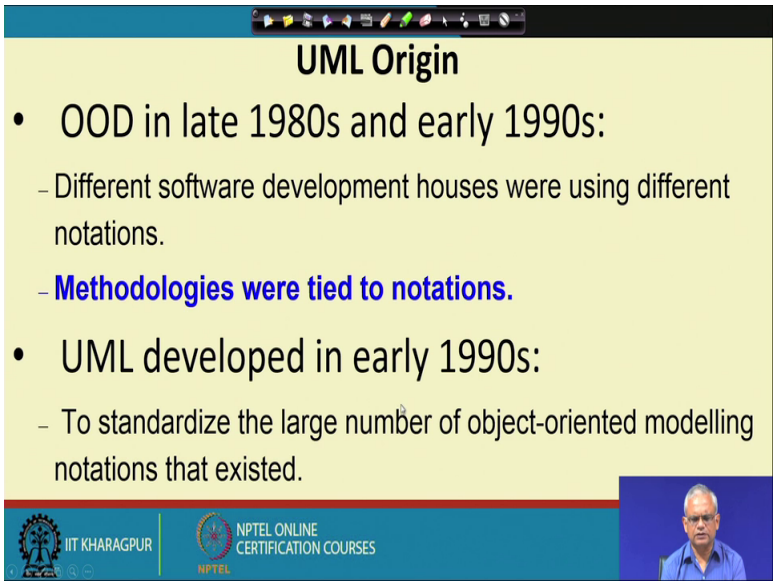


As the name says UML stands for unified modelling language.

And therefore, UML stands for unified modelling language and as the name implies it is a modelling language or we can say that it is a language, UML is a language using which we can document the model and the design. Let us be clear that UML is just a language, when we learn a language we can document things and it is not a design and development methodology. It is just a language we need to learn the language and we will also discuss a methodology for designing and we use UML as the language to document the design, but the design will obtained using a design methodology, and we use UML as a documentation for that design.

One good thing about the UML is that it is independent of any design methodology. UML is used for documenting the design, but the design can be obtained using any methodology we can always document using UML.

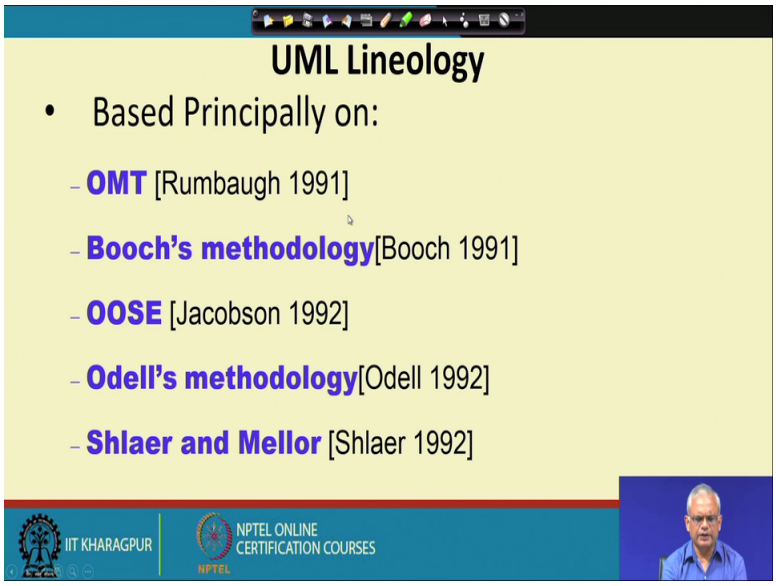
(Refer Slide Time: 03:41)



The, UML let us see how it came about as we are saying that the object oriented design was first used in 1980s and when the object oriented design came up, different researchers, they proposed different methodologies. There were a large number of methodologies for object oriented design in the 1980s and 90s. And, one thing about this methodologies is that each of them they use different notations. And therefore, it became very difficult in companies or for the students, because in a company there may be many projects and each project was maybe using, different methodology they are documenting the result in different notations.

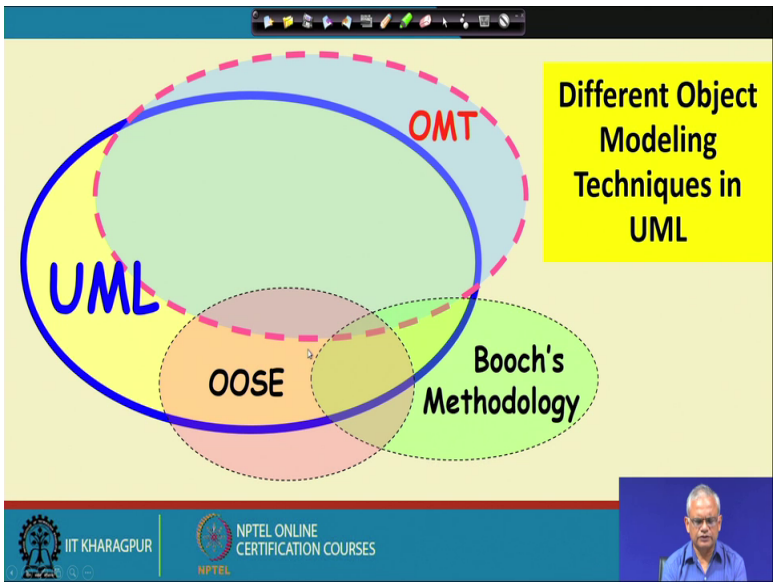
And therefore, one project cannot really reuse or understand the design of another team and it led to lot of confusion. It was felt that there needs to be standardization and in the early 90s an attempt was made to standardize all these different design methodologies and notations.

(Refer Slide Time: 05:14)



And, those days the ones that were extremely popular were the OMT object modelling technique a Rumbaugh, Booch's methodology by Grady Booch, object oriented software engineering by Jacobson, Odell's methodology by Odell Shlaer and Mellor methodology. And, each of this methodology there was a design methodology and also there were very different notations. And when there was a effort to standardize these methodologies. Obviously, they had to look at these methodologies and notations that they were following and based on that the UML was proposed.

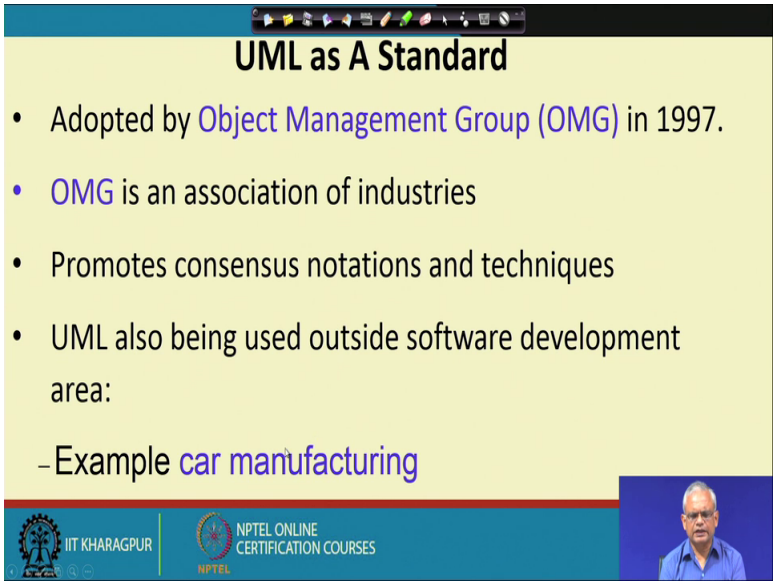
(Refer Slide Time: 06:01)



As, you can see in this diagram that the notations for the UML have been borrowed largely from the OMT, which was extremely popular design technology having it is own notations. Some of the notations are taken from Booch's methodology, some of the notations are taken from the object oriented software engineering and some of the methodologies sorry some of the notations were not there in any of these methodology and these are new.

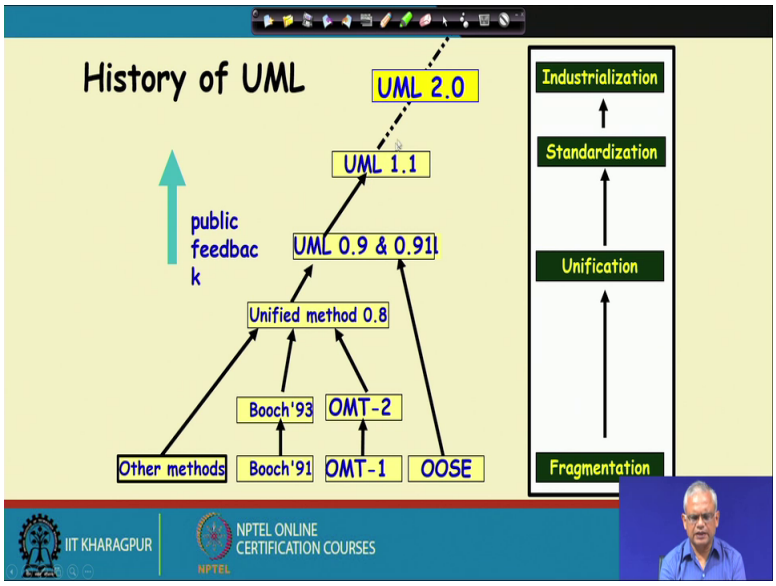
We look at all these notations as a part of our discussion and UML, but then we must remember that most of the notations here are borrowed from some of the popular methodologies. And, it had of course, few of the notations on it is own as we proceed, we will point out from where these methodologies sorry these notations have been taken.

(Refer Slide Time: 07:13)



The object management group it adopted OMG, OMG sorry the OMG adopted the UML the object management group adopted the UML in 1997 and OMG is an association of industries. It tries to promote consensus notation and techniques just to have uniformity standardization, but this is not really a standardization body it can just adopt it is own notations. So, that become popular and ultimately need to lead to standardization. And, once it was adopted by the OMG became extremely popular and not only it is used extensively by the software development area, but also even domains which are entirely different, they are also using UML notations.

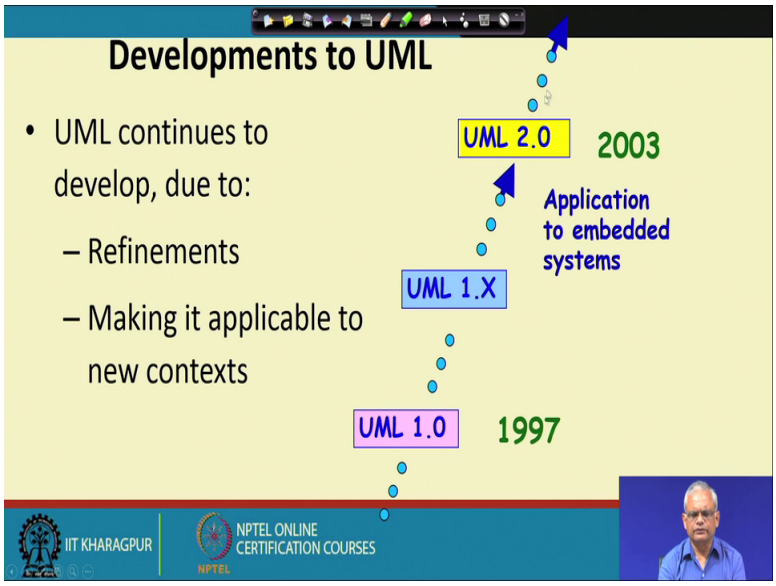
(Refer Slide Time: 08:21)



If, we look at the development of UML we can see that various methodologies were combined here and finally, the UML version one was released in 1997. So, there were a fragmentation of different methodologies in the 80s and 90s and towards the end of nineties is your unified into the unified modelling language. And, then as these were being used there were need for extensions to the UML.

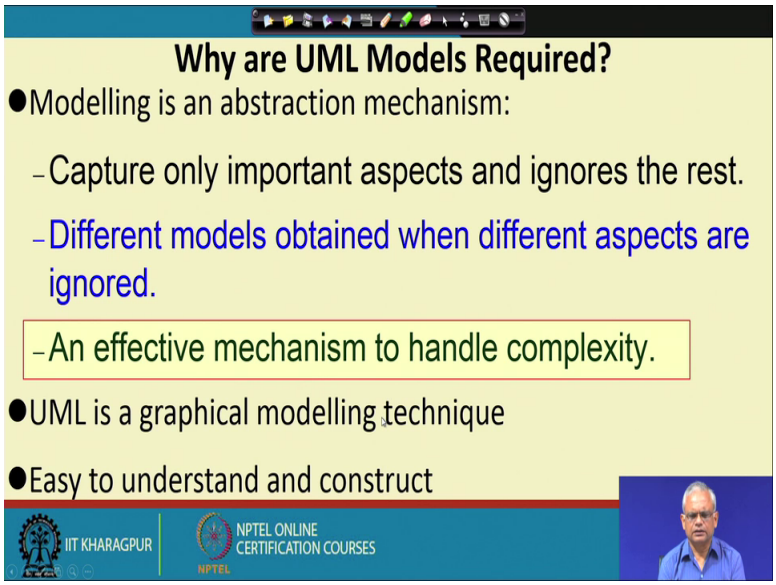
For example, to make it use in a particular domain may be some more notations are needed and so on. So, different versions of UML came up and finally, as these were applied to different domains, there were some shortcomings of the UML were notated. For example, it does not handle events parallel processing and so on. And, UML 2.0 was released in the year 2004 and mainly to make it applicable to some domains of industry like embedded systems and so on.

(Refer Slide Time: 09:47)



So, UML 1.0 released in 1997 it was a unification of various methodologies that existed. And, then it continued to evolve until 2003 the UML 2.0 version was released and even now it continues to evolve further.

(Refer Slide Time: 10:11)



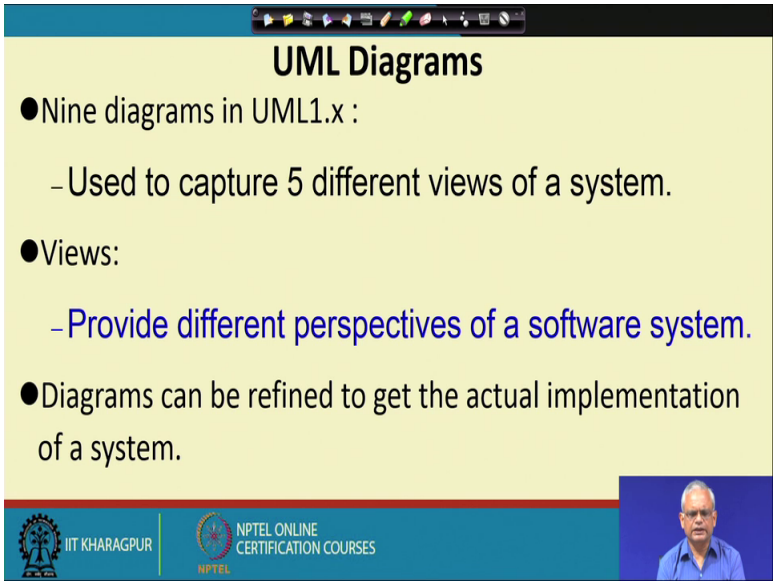
First thing let us understand before we discuss about UML modeling, that why do we need a model. Also, we need to answer the question is modelling the same as designing. If, you remember the early parts of our discussion where said that models are actually an abstraction mechanism abstraction is necessary to simplify complex problems. We need

to create an abstraction where we develop a simple model of a complex problem and then slowly we through various hierarchies we have the entire problem modeled.

And, then once we model this problem we can design from the model. So, model is a abstraction mechanism where we construct a very simple representation of the problem, by ignoring different aspects of the complex problem. And, it is an effective mechanism to handle complexity typically we use a graphical modelling technique, and this is called as the analysis model, where we model the problem domain.

And, once we come up with the analysis model we can convert it to a design model. So, to answer the question that is designing the same thing as modelling yes design is a model, but all models are not designs. We can have analysis models which are just models of the problem and the model of the problem can be transformed into a design model, which can be easily implemented.

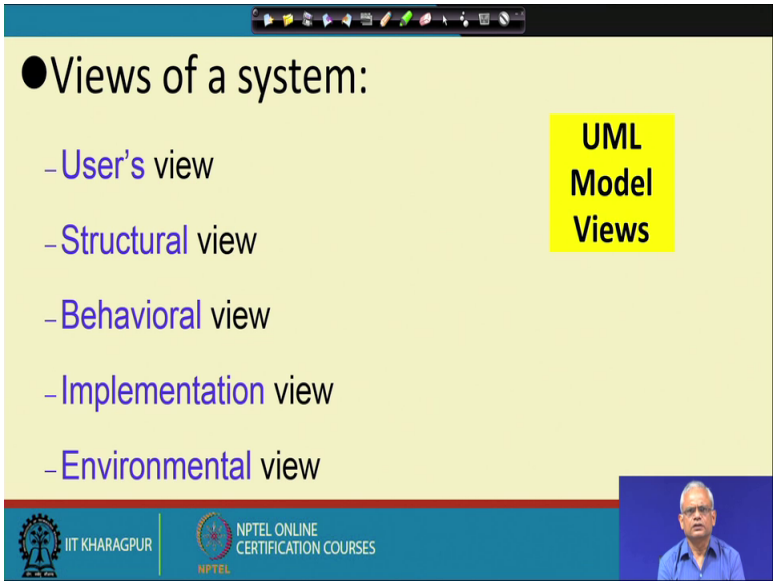
(Refer Slide Time: 12:19)



Now, let see what are the diagrams that are used in UML, because we said that UML is a language and it is a graphical language in this graphical language we have different diagrams. We will see the graphical notation that is the vocabulary of the UML and how to construct the model? There are 5 views of a system and these views provide different perspectives of software.

And, then we develop this UML diagrams and then we transform it or refine it to get the actual design model and the implementation of the system.

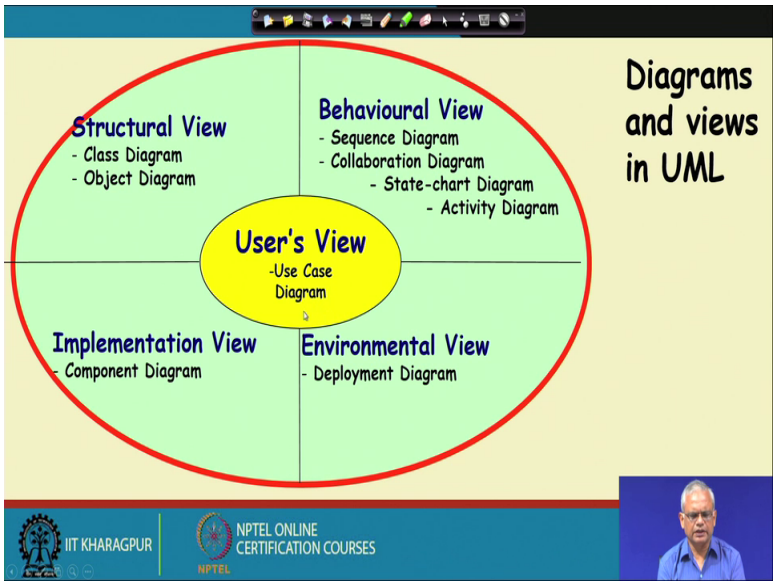
(Refer Slide Time: 13:09)



The 5 views of the system are the users view, structural view, behavioral view, implementation view and the environmental view. The users view is the one which says how does the customer or the user view the system? So, this is the external view. The structural view is a internal view of the software. Namely, what are the classes? How are the classes related? What are the objects object relations and so on. Whereas a behavioral view, it represents once a invocation or a input from the user is obtained, which part of the internal structure.

They behave in what way and finally, produce the output. The implementation view this discusses, how are the different elements of the internal structure organized and the environmental view says it depicts? How the different parts of the system? How are they deployed?

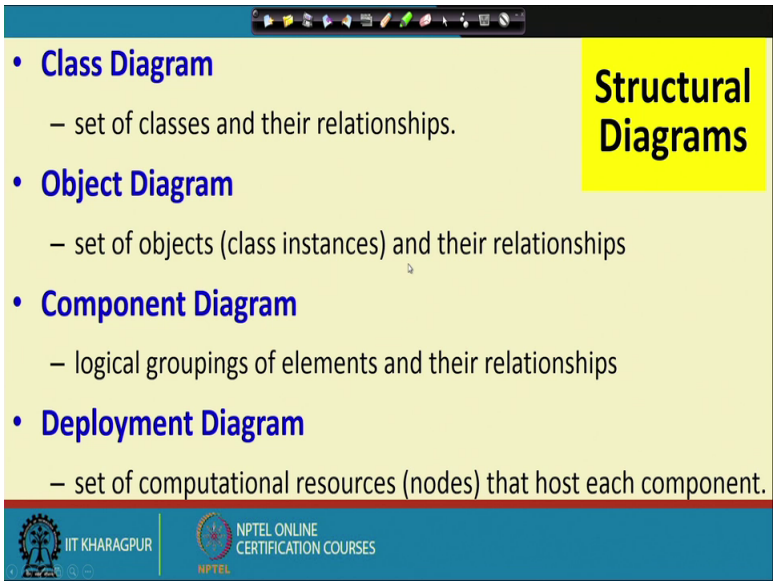
(Refer Slide Time: 14:28)



If, we represent that here the users view this is the view of the user and it is represented in the form of a use case diagram. We have drawn it at the centre because this is the central view based on which other are developed other views and other diagrams are developed. The users view is the starting point, because the user gives the requirements. And, the requirements are modeled using a use case diagram and based on this the other diagrams are developed, other views and diagrams and that is why we have drawn this at the centre it is the central model of any software.

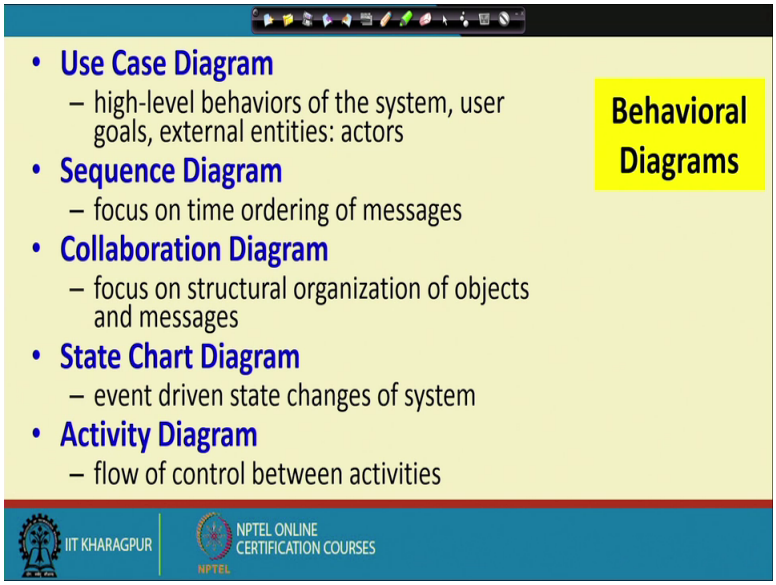
The structural view are represented by the class diagram and the object diagram, the behavioral view through sequence diagram, collaboration diagrams, state chart diagram, and the activity diagram. And, the environmental view through deployment diagram, the implementation view through the component diagram.

(Refer Slide Time: 15:36)



Now, let us see the structural diagrams. So, this consists of the class diagram which are set of classes and their relationships, the object diagrams which are set of objects and their relations, the component diagrams which are a logical grouping of elements and their relations, the deployment diagram which are where these are hosted.

(Refer Slide Time: 16:01)



Now, the behavioral diagrams are the use case diagram, the sequence diagram, collaboration diagram, the state chart diagram, and the activity diagram.

The use case diagram are the high level behavior of the system; the user goal external entities actors. Whereas, the sequence diagram it depicts the focus on how the messages are exchanged.