Object-Oriented Analysis and Design

Prof. Partha Pratim Das
Department of Computer Science and Engineering
Indian Institute of Technology - Kharagpur

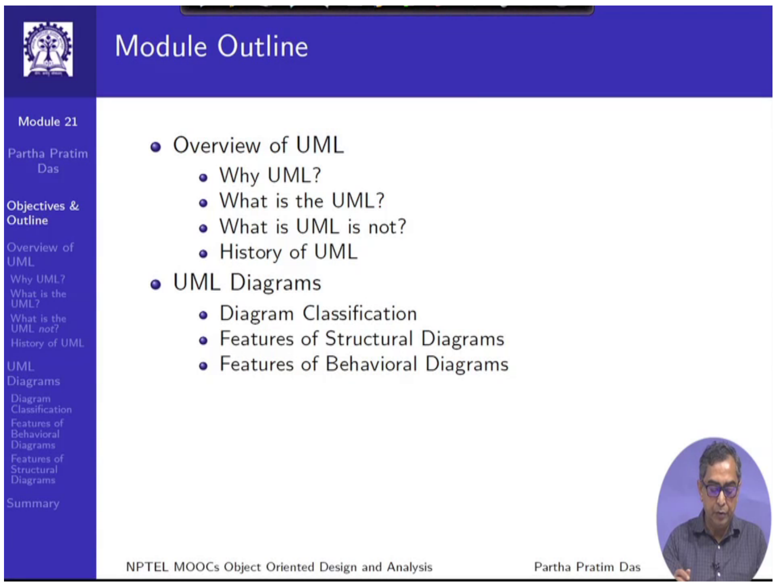
Lecture - 33
Overview of UML

Welcome to module 21 of Object Oriented Analysis and Design in the last couple of modules, we have discussed and actually worked out an exercise with a Leave Management System to illustrate how different linguistic and important analysis techniques can be engaged to extract the classes the key abstractions their relationships and so on.

In this context we can go further and that is our endeavor to do so but before we proceed further on extracting more and more system information from the specification and start modelling them for proper design we need a vehicle to express the information that we are extracting, naturally we are our starting point has been certain English documents interaction videos and it could be other supporting documents images charts and so on.

But as we extract the information and start creating the analytical design of the system we need a language which can be used very compact way and very concretely to express all the information that we are extracting. UML or Unified Modelling Language is such a vehicle so in the module today and in several modules that are to follow this.

And in the next couple of weeks we will take a deep look into what is UML, how UML is represented and as we work through different representations of system concepts designs parameters into the UML we will illustrate as an ongoing process as to how we can use that Leave Management System and create its Unified Modelling Language models so with that a broad objective we will start this module by discussing it basic overview of UML.
(Refer Slide Time: 03:17)



Module Outline

Module 21

Partha Pratim Das

Objectives & Outline

Overview of UML

Why UML?

What is the UML?

What is the UML *not*?

History of UML

UML Diagrams

Diagram Classification

Features of Behavioral Diagrams

Features of Structural Diagrams

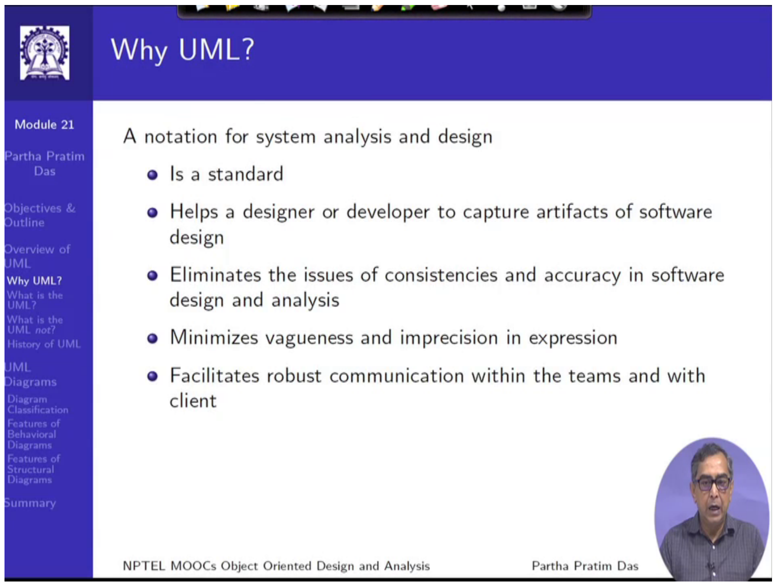
Summary

- Overview of UML
  - Why UML?
  - What is the UML?
  - What is UML is not?
  - History of UML
- UML Diagrams
  - Diagram Classification
  - Features of Structural Diagrams
  - Features of Behavioral Diagrams

NPTEL MOOCs Object Oriented Design and Analysis   Partha Pratim Das

This is what we intend to discuss and as usual this outline would be available to the left of every slide as we go through the presentation.

(Refer Slide Time: 03:27)



Why UML?

Module 21

Partha Pratim Das

Objectives & Outline

Overview of UML

Why UML?

What is the UML?

What is the UML *not*?

History of UML

UML Diagrams

Diagram Classification

Features of Behavioral Diagrams

Features of Structural Diagrams

Summary

A notation for system analysis and design

- Is a standard
- Helps a designer or developer to capture artifacts of software design
- Eliminates the issues of consistencies and accuracy in software design and analysis
- Minimizes vagueness and imprecision in expression
- Facilitates robust communication within the teams and with client

NPTEL MOOCs Object Oriented Design and Analysis   Partha Pratim Das

So the first question certainly is why do you need something like UML, as we have seen that so far there are two kinds of descriptors that we have for a complex system on one side on one extreme is a descriptor which specifies the requirements of the system specifies different behaviour and entities of the system and typically as we have observed it is written in a natural language.

And as we have repeatedly explained there could be supporting information discussions, notes, images and all that the key difficulty in this form of representation of a system is natural language is often ambiguous vague imprecise from the same text what I understand what you understand and what someone else will understand would vary widely the diagrams help and yet they could be drawn with different conventions with different practices.

So this way of representing a system is not a uniquely method by which we can unambiguously precisely understand what the system is all about, the extreme of representing a system is its implementation when the system is gets implemented in terms of possibly a Object Oriented Language so if we intend to use C++ or Java are some such, then we will have the whole system again described represented in terms of a set of classes methods exception conditions dependencies modules of sub systems messages threads and on so forth.

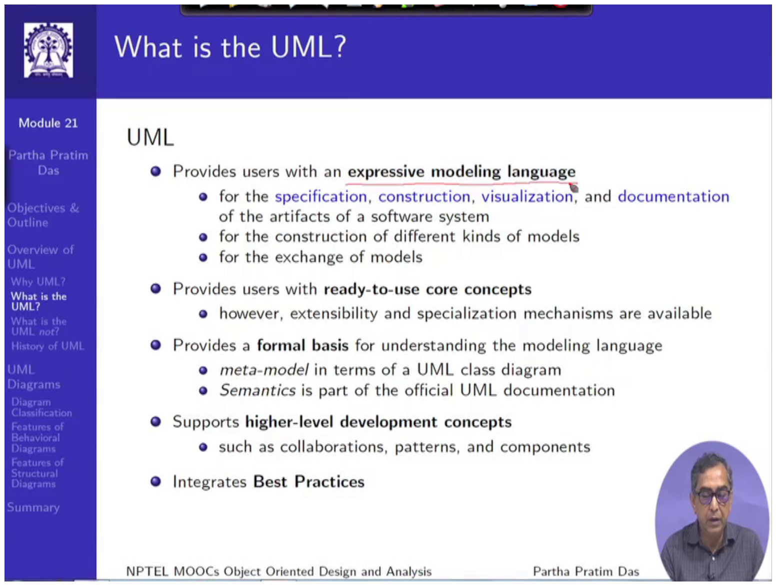
Again another description which is very specific and would vary across different platforms different languages different paradigms and so on, the our intention our requirement in the whole process of OOAD is to start with a requirement specification of a system and reach the implementation software implementation of the system both of them we find a difficult complex to handle they are not very clear precise in the expression.

So that brings in the context of using something like a UML or Unified Modelling Language which is basically a notation for system analysis and design the first importance of why should you use UML it is a standard it is an International standard maintained by the object modelling group or OMG.

So it helps the designer developer even the user to capture various artifacts of software design various inventions details of software design can be uniquely captured in UML without any scope for imprecision vagueness ambiguity or the viewer or reader dependent understanding so it eliminates the issues of consistencies and accuracy that typically will exist in a software analysis and design problem it minimizes vagueness and imprecision.

As I mentioned and coupled with all these it facilitates a robust communication with within teams robust communication between the vendor and the client and so on, so the basic process would be that the initial stages of our design and analysis we will try to interpret the specification from different multiple natural language and multimedia sources and create this UML diagrams which will be the unique representation and kind of the golden on which the whole system will then be built up.

(Refer Slide Time: 08:01)



What is the UML?

UML

- Provides users with an **expressive modeling language**
  - for the specification, construction, visualization, and documentation of the artifacts of a software system
  - for the construction of different kinds of models
  - for the exchange of models
- Provides users with **ready-to-use core concepts**
  - however, extensibility and specialization mechanisms are available
- Provides a **formal basis** for understanding the modeling language
  - *meta-model* in terms of a UML class diagram
  - *Semantics* is part of the official UML documentation
- Supports **higher-level development concepts**
  - such as collaborations, patterns, and components
- Integrates **Best Practices**

So what is UML, UML has can we look at in various dimensions first of all its an expressive modelling language it is a language to model its set to be a expressive language because it is allows you to model almost anything that you need to model starting from specification, to construction, to visualization, to documentation everything that you need in the whole life cycle of a software development system implementation development can be captured in the UML.

For the need of clarity expressiveness and precision UML is almost totally designed as a graphical language so it is not which depends either on the vague semantics of natural language nor does it deal with the complex logic and analysis of the programming languages it is a graphical pictorial kind of language and it allows us to use a whole lot of ready-to-use core concepts.

You recall from the earlier discussion that we had talked about reuse we're talked about common patterns that occur frequently in complex systems that occur across domains and so on, so what UML has done UML has tried to create semantic expressions for those concepts and met them available in the language itself so that very often once you come across such a concept in the system you are going to represent you would simply be able to find a primitive find a feature in the UML which directly represents that.

Besides that, UML provides a formal basis may be for some of you it may be difficult to understand what a formal basis means what it means is though it is a graphical language it should not be just taken as a pictorial representation it has certain strong mathematical foundation which can be proved to always provide a consistent correct description of the system alone, and in that process it uses certain meta-model it uses a semantics which is the part of the official UML standard documentation.

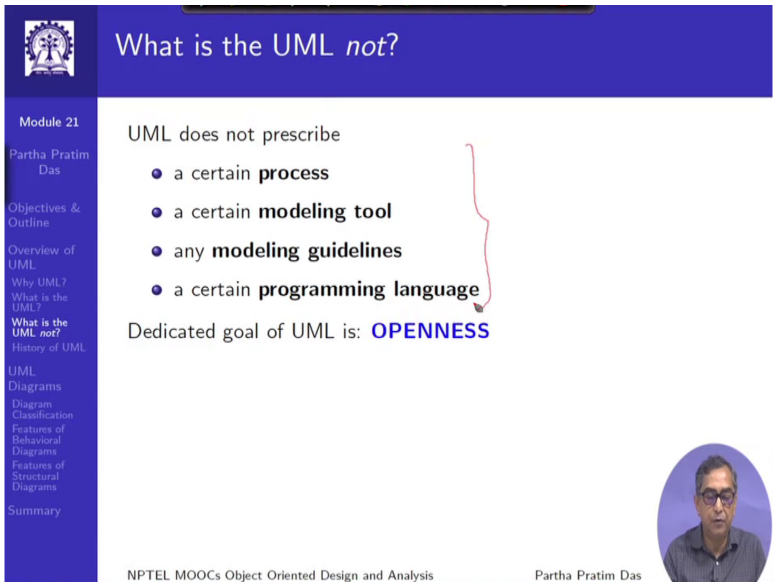
In addition in the supportive side it supports high level development concepts will often find UML talking about terms and items which can be directly represented and immediately implemented in an standard OOP language like C++ some of those UML concepts a supported concepts will actually have given standard library implementation in some of the common OOP languages this kind of collaborations patterns these are the some of the terms that I have mentioned here patterns and components would be very convenient to represent in UML.

Last but not the least is UML inculcate a set of best practices if you refer back to the very first module where we started discussing as to what at the challenges of constructing a software we address that there are different issues and maturity issues and so on, for which we still set the fact software is developed but all other kinds of engineering development is actually called construction whether there is a guarantee of success.

We had mentioned that all these other fields of engineering succeed so well are so reliable these days because of the years and years of practice we have been able to develop a set of norms set of industry standards and best practices which every civil engineering design or a mechanical machine design has to follow.

Similar practices still do not exist that much in software so UML can be used as a vehicle encode this best practices because the UML is giving as a platform in which I can express a concept in a platform independent language, language independent even at times domain independent manner, so this is a basic essence of UML.

(Refer Slide Time: 12:45)



What is the UML *not*?

UML does not prescribe

- a certain **process**
- a certain **modeling tool**
- any **modeling guidelines**
- a certain **programming language**

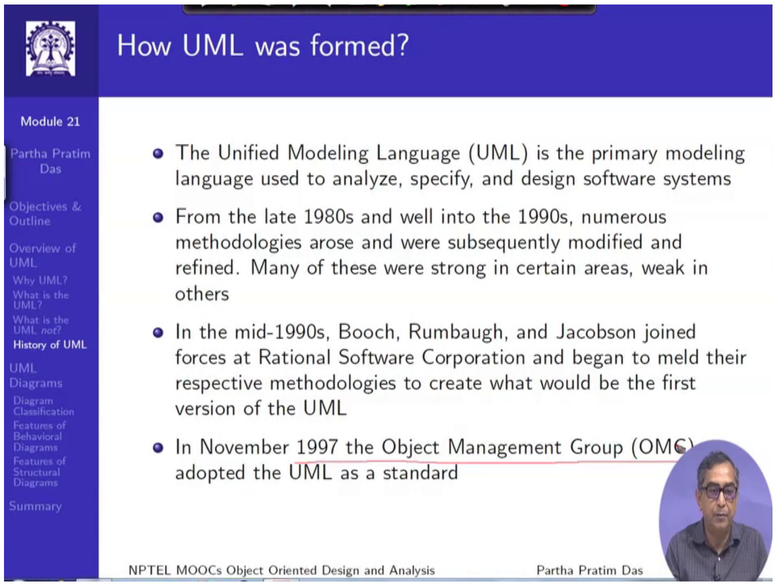
Dedicated goal of UML is: **OPENNESS**

And while we discuss this it is also important to understand and highlight that UML is a - is characterized by Openness, what does that mean that UML does not say that you have to use a specific modelling tool or you have to use a specific language it does no - it would not say that you must use C++ or you must use Java or you must use python.

It will not tell you that this is you should use the waterfall process of development or agile process of development or you should use industrial development it will not make any prescription it is not meant for prescribing any of this or any of the other actual detail practices that are involved.

It is open so that irrespective of what process you follow, what modelling language, what programming language, what platform, which hardware you are planning to use irrespective of all these parameters UML would be equally useful for you to model the system and to take it through the life cycle of development.

(Refer Slide Time: 13:55)



How UML was formed?

Module 21

Partha Pratim Das

Objectives & Outline

Overview of UML

Why UML?

What is the UML?

What is the UML not?

History of UML

UML Diagrams

Diagram Classification

Features of Behavioral Diagrams

Features of Structural Diagrams

Summary

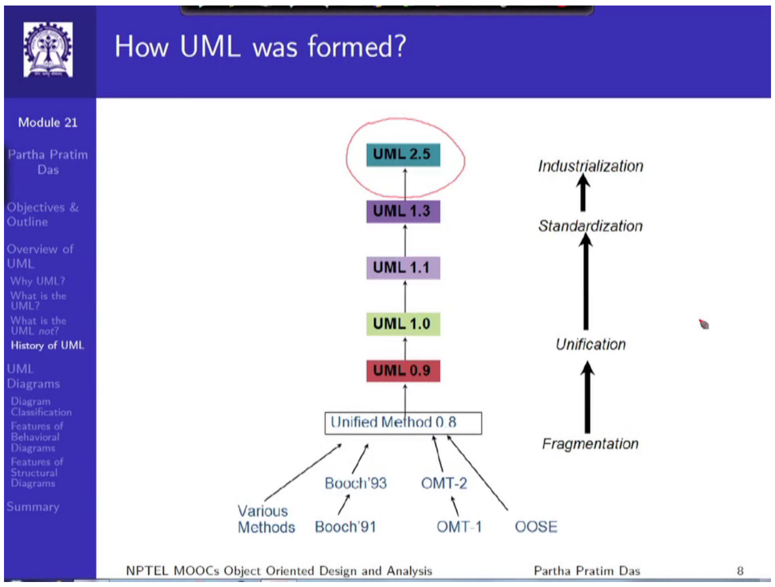
- The Unified Modeling Language (UML) is the primary modeling language used to analyze, specify, and design software systems
- From the late 1980s and well into the 1990s, numerous methodologies arose and were subsequently modified and refined. Many of these were strong in certain areas, weak in others
- In the mid-1990s, Booch, Rumbaugh, and Jacobson joined forces at Rational Software Corporation and began to meld their respective methodologies to create what would be the first version of the UML
- In November 1997 the Object Management Group (OMG) adopted the UML as a standard

NPTEL MOOCs Object Oriented Design and Analysis Partha Pratim Das

So a quick bit in terms of how UML got formed I really do not have the time to go over the details of history but that history is indeed very interesting, the concept of being able to model system in a formal way started quite long back and in 80s show a lot of activities and which went into at least the first part of the 90s and it saw a proliferation of several different ways several different languages and protocols to model systems, till majority of the proponents of various models Booch, Rumbaugh, Jacobson and so on.

You - you are already familiar with Booch's name we are following his book very thoroughly they all collaborative join hands and UML was formally that way created with the formation of the Object Management Group there is a website on OMG I would suggest everybody that you go OMG site and you will find a whole lot of very useful information not only on UML but in general on Object Oriented Analysis and Design, and especially you will find information which is oriented towards a current practices.

(Refer Slide Time: 15:21)



How UML was formed?

Module 21

Partha Pratim Das

Objectives & Outline

Overview of UML
- Why UML?
- What is the UML?
- What is the UML *not*?
- **History of UML**

UML Diagrams
- Diagram Classification
- Features of Behavioral Diagrams
- Features of Structural Diagrams

Summary

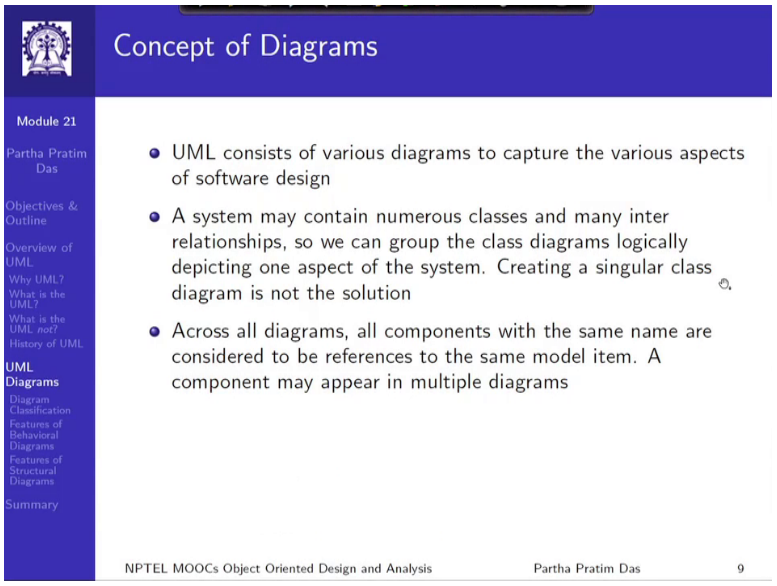
NPTEL MOOCs Object Oriented Design and Analysis Partha Pratim Das 8

So in this background this kind of late 90s that UML started existing in its current flavor so this is - this is kind of the era of fragmented development then people has joined hands and stay put all this together and the U in UML that is in Unified in UML is actually the unification of all these methods you know some people tend to thank that it is called unified because it can be used over multiple faces of SDLC are used to express variety of concepts.

Those are also true but in primarily got its name by the unification of methodologies that it represented in the late 90s. And then over this whole around 25 years nearly 30 years I am sorry its nearly 20 years it has developed and OMG managers and the versions UML from one to the other and this UML 2.5 is a current version that is existing in the standard. So while we discuss UML, we will over discuss some of the starting from the early concept.

So not we will often restrict ourselves to the 1.X kind of standard because 2.5 has lot of advanced concepts which we may not have time for but we will take glimpses of 2.5 from time to time as it needs.

(Refer Slide Time: 16:54)



Concept of Diagrams

- UML consists of various diagrams to capture the various aspects of software design
- A system may contain numerous classes and many inter relationships, so we can group the class diagrams logically depicting one aspect of the system. Creating a singular class diagram is not the solution
- Across all diagrams, all components with the same name are considered to be references to the same model item. A component may appear in multiple diagrams

NPTEL MOOCs Object Oriented Design and Analysis Partha Pratim Das 9

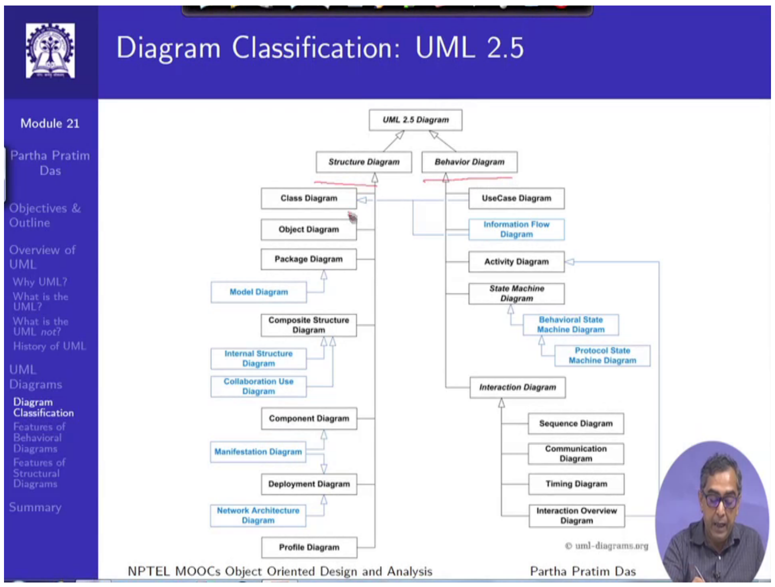
As I said the UML is graphical so what it has the core of UML is collection of diagrams so it consists of previous diagrams that captured various aspects of software design, so the idea is simple that a software system or a system in general we have seen that there several different complexities and there are several different ways that we can look at the system we have discussed these things at length in the earlier modules.

So the UML takes a very simple approach to address this, that it has identified what are the major aspects of that you need to know that you need to decide on for a complex system and for each one of them there is a category of diagrams and that diagram has specific primitives which help you actually capture as well as analyze the aspects of that.

So for example you will have a bunch of class diagrams necessarily for - we have already seen that is not easy to capture all classes in to one set so a system will have multiple classes, multiple kinds of relationships so the class diagram can be built up over this collection into smaller to bigger set to give you a very unique UML description.

And it has UML supports a strongly support a name unification which means that across different diagrams and across different modules of the design the same name can be used and therefore we will see that a name that is used in this use case to represent a use case will be used in the class diagram to represent a method the corresponding method, a an actor in a use case will be become a class in the class diagram representing a particular group of individuals is taking that action so that's the - that's the broad basic point.

(Refer Slide Time: 19:09)



And based on that this is just to give you an overview glimpse of the whole thing so you can see that, the route is the UML 2.5 diagram the collection of all diagrams and then you have two types of diagrams the Structure diagram and Behavior diagram.

And as you must have noticed by now is whenever we draw this kind of an arrow let me draw it clearly again that's an open triangle with this then which means a is a or generalization specialization relationship so to describe the UML diagrams we are taking care of the same notation, this notation is part of the UML class diagram and some of the other UML diagrams so we say structure diagram is UML 2.5 diagram this arrow.

The behavior diagram is a UML 2.5 diagram this arrow and so on, so basically here in this slide not only see a collection of these diagrams but you see kind of there specialization generalization hierarchy so we know that these are the two basic types of diagrams and then we have a variety of different class structure diagrams like this, there variety of different behavioral diagrams like this and so on.

In reading this chart you have to remember a basic color code the diagrams that are written in black are part of the standard so Class diagram or Use case diagram are part of the UML 2.5 standard, but the diagrams that are written in blue are the level of proposals they are being discussed and they have not yet made to the standard some of them may become a part of the standard UML in future some of them may get dropped also and there are several inter-relationships between them.

For example, if you are talking about a composite structure diagram then you are saying that there are two specific sub types of it in terms of the internal structure and the collaboration use diagram so these are at the level of proposal so what we will do this I am presenting you the whole picture to just give you the current truth of what the UML 2.5 looks like.

But we will concentrate primarily on the diagrams in the standard and there too we will just take up a subset because it may be otherwise quite a lot of time taking to complete all of the diagrams we will just take a measure ones and discuss this.
(Refer Slide Time: 22:03)

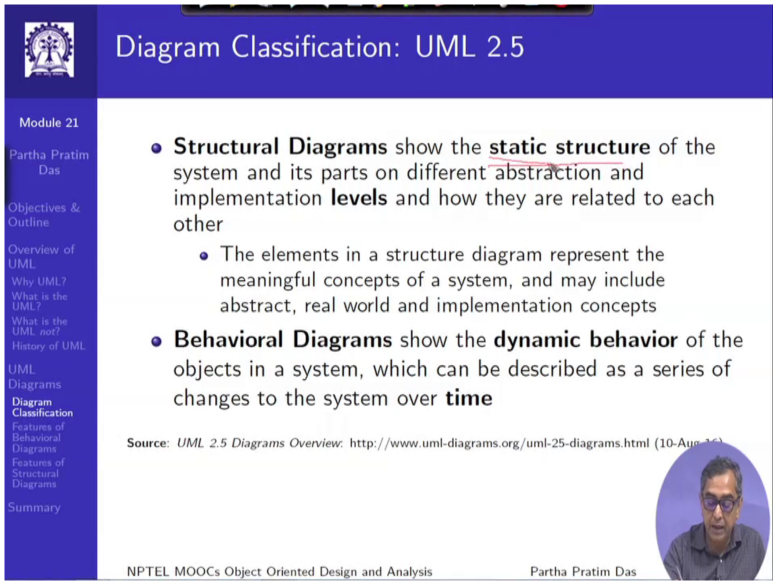


Diagram Classification: UML 2.5

- **Structural Diagrams** show the **static structure** of the system and its parts on different abstraction and implementation **levels** and how they are related to each other
  - The elements in a structure diagram represent the meaningful concepts of a system, and may include abstract, real world and implementation concepts
- **Behavioral Diagrams** show the **dynamic behavior** of the objects in a system, which can be described as a series of changes to the system over **time**

Source: *UML 2.5 Diagrams Overview*: http://www.uml-diagrams.org/uml-25-diagrams.html (10-Aug-16)

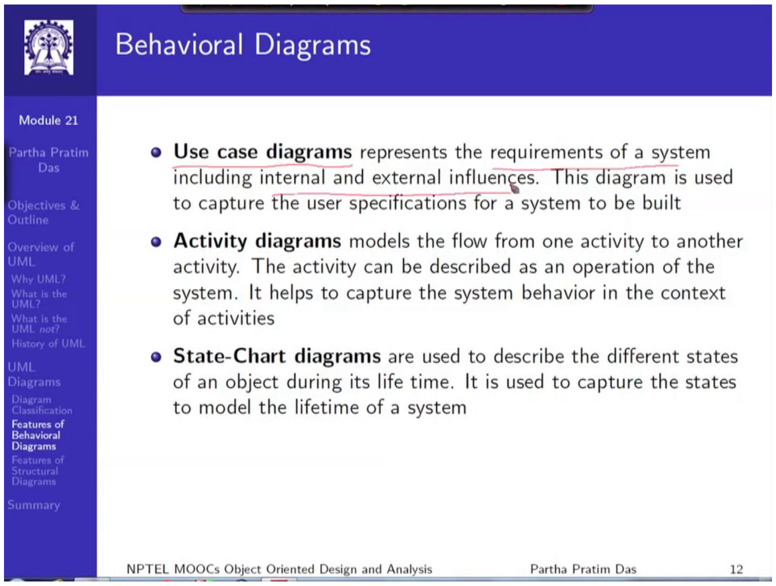
So first let me say a few words about the basic tool sub classes of the UML diagrams, one is called Structural diagrams and the structural diagram show the static structure of the system so we have had we have been seeing this notion of static and dynamic description all through I mean - I would refer you back to our discussions on objects and their nature we observed that objects have an identity but besides that they have attributes and they have properties, attributes are the static structure and properties are the dynamic behaviour.

So similarly the diagrams here are also in a way static structural diagrams give us a static structure, that is how different parts of the abstraction at different implementation level are related to each other that, but this is static which means that when the system is actually built up deployed and works these are not going to change over time these are - this remind kind of fixed.

The other side behavioral diagram which talks about the dynamic behavior that is how does the different objects the behavior of different objects change with time what happens if I want to get into a system and I want to perform a login I have to perform a sign in kind of activity then what are the different actions that into happen in what sequence, in what temporal order, in with what dependencies to allow or dis allow me to sign into the system is a part of the behavioral or the dynamic behavior.

So if we cannot that some of the subset of the diagrams we will talk about structural properties and subset we'll talk about behavioral properties.

(Refer Slide Time: 24:11)



Behavioral Diagrams

- **Use case diagrams** represents the requirements of a system including internal and external influences. This diagram is used to capture the user specifications for a system to be built
- **Activity diagrams** models the flow from one activity to another activity. The activity can be described as an operation of the system. It helps to capture the system behavior in the context of activities
- **State-Chart diagrams** are used to describe the different states of an object during its life time. It is used to capture the states to model the lifetime of a system

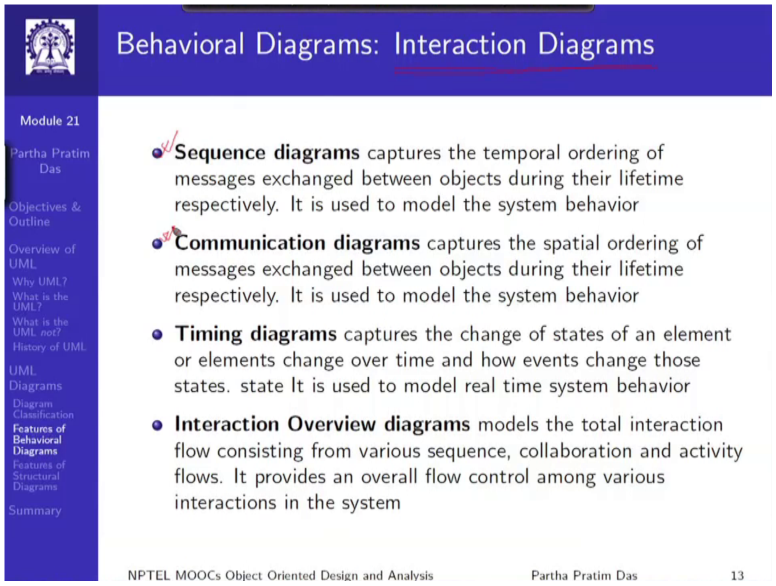
So let us start just take a very quick look in terms of this so the - the way of organized is for all the diagrams exist in the standard we are first talked about the behavioral diagrams and then we talk about the structural diagrams. Here I am just trying to make a very brief overview to you about this each and every diagrams we will take up many of them at length and we will have separate modules discussing regard at - at times multiple modules discussing them how to use that wherever applicable.

We will take the case of the Leave Management System and illustrate how the corresponding diagrams can be met up. So the most important possibly to start with and the first fundamental diagram that UML behavioral has to start is Use case diagram where you represent the requirements of a system including the internal and external influences what is happening around the system there many gross ways.

Then Activity diagram talks about what are the different activities that internally happens in the system what are the operations of the system. State-chart diagram basically borrows its mean from the state transition machine or finite-state automaton are finite state machine so it means that I can think of that in the system there are several situations where there are a finite set of possibilities and the system keeps on moving from one possibility to the other and that is what we say is the finite state machine.

State-chart basically is an enhancement of the finite state machine which along with telling me what are the different states that a system can be in for example in the sign on system I could be I may have signed up and I may not have signed up if I have signed up then I'm signed in or I am not signed in if I have signed in then I may have a state of coming in next time without having to sign in again and so on.

So these are the different states and state-chart that allows me to play around with these different state information.
(Refer Slide Time: 26:17)



Behavioral Diagrams: Interaction Diagrams

Module 21

Partha Pratim Das

Objectives & Outline

Overview of UML

Why UML?

What is the UML?

What is the UML *not*?

History of UML

UML Diagrams

Diagram Classification

Features of Behavioral Diagrams

Features of Structural Diagrams

Summary

- **Sequence diagrams** captures the temporal ordering of messages exchanged between objects during their lifetime respectively. It is used to model the system behavior
- **Communication diagrams** captures the spatial ordering of messages exchanged between objects during their lifetime respectively. It is used to model the system behavior
- **Timing diagrams** captures the change of states of an element or elements change over time and how events change those states. state It is used to model real time system behavior
- **Interaction Overview diagrams** models the total interaction flow consisting from various sequence, collaboration and activity flows. It provides an overall flow control among various interactions in the system

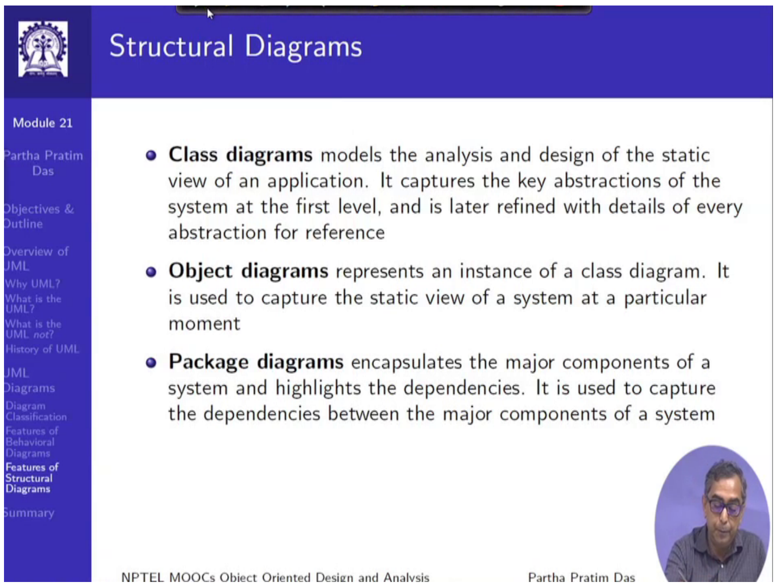
NPTEL MOOCs Object Oriented Design and Analysis Partha Pratim Das 13

The next set of behavioral diagrams are all clubbed in to one sub category known as interaction diagram because the define different kinds of interaction behaviour other the Sequence diagram is most frequently used which will tell you what is the order of events that happened. The Communication diagram will tell us that what is the order in which objects exchange messages I would just again remind you that our vehicle finally is a client server model.

So when you do all these modelling finally we would like to take it to some messages being sent from one object to the other so the sequence diagram tells us the temporal order in which the messages are sent, and the communication diagram tell us the special order or the object to object order of how messages are sent. Timing diagram as the name suggest is closely related to the timing information and Interaction Overview diagram basically tries to model that how interaction flow be from various sequences collaborations and activity flows.

So it is kind of sets little bit at a higher level compare to the sequence, communication and timing diagrams which are particularly object based, interaction overview diagrams is often at a level of modules.

(Refer Slide Time: 27:41)



Structural Diagrams

Module 21

Partha Pratim Das

Objectives & Outline

Overview of UML

Why UML?

What is the UML?

What is the UML *not*?

History of UML

UML Diagrams

Diagram Classification

Features of Behavioral Diagrams

Features of Structural Diagrams

Summary

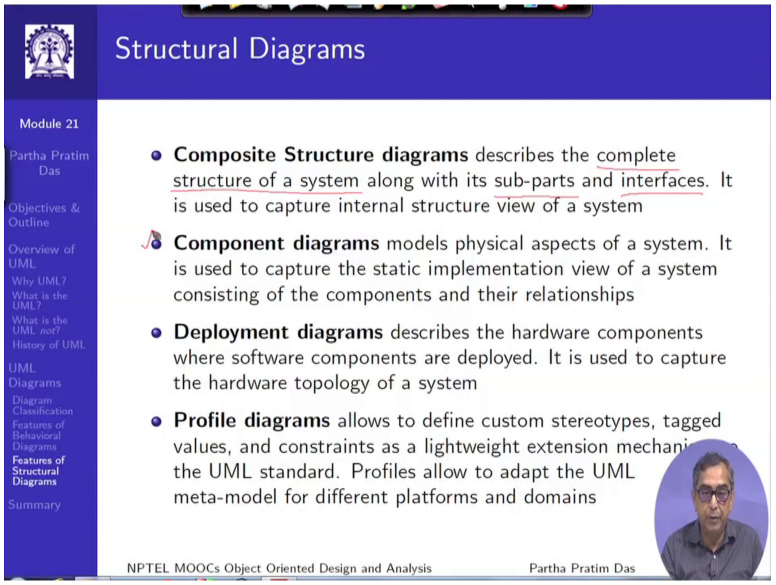
- **Class diagrams** models the analysis and design of the static view of an application. It captures the key abstractions of the system at the first level, and is later refined with details of every abstraction for reference
- **Object diagrams** represents an instance of a class diagram. It is used to capture the static view of a system at a particular moment
- **Package diagrams** encapsulates the major components of a system and highlights the dependencies. It is used to capture the dependencies between the major components of a system

NPTEL MOOCs Object Oriented Design and Analysis Partha Pratim Das

Coming to the structural diagrams the main structural diagram of course are the Class diagrams and Object diagrams, class diagrams and as we go into detailing them after a couple of modules we will find that you have already come across much of the notation are the diagram convention of the class diagrams and the objects in terms of our process of identifying classes and our defining objects.

Certainly the other structural diagrams Package which basically encapsulates the different components and puts them as a part of a bigger module.

(Refer Slide Time: 28:19)



Structural Diagrams

Module 21

Partha Pratim Das

Objectives & Outline

Overview of UML

Why UML?

What is the UML?

What is the UML *not*?

History of UML

UML Diagrams

Diagram Classification

Features of Behavioral Diagrams

Features of Structural Diagrams

Summary

- **Composite Structure diagrams** describes the complete structure of a system along with its sub-parts and interfaces. It is used to capture internal structure view of a system
- **Component diagrams** models physical aspects of a system. It is used to capture the static implementation view of a system consisting of the components and their relationships
- **Deployment diagrams** describes the hardware components where software components are deployed. It is used to capture the hardware topology of a system
- **Profile diagrams** allows to define custom stereotypes, tagged values, and constraints as a lightweight extension mechani[illegible] the UML standard. Profiles allow to adapt the UML meta-model for different platforms and domains

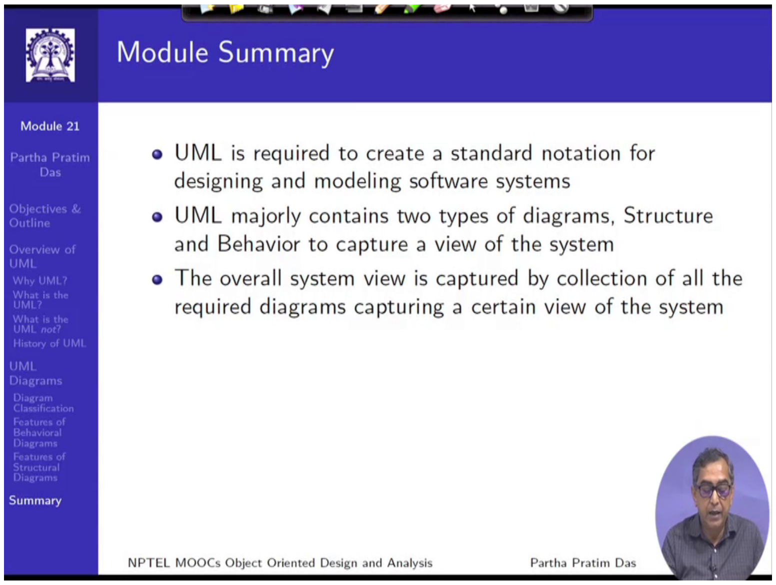
NPTEL MOOCs Object Oriented Design and Analysis Partha Pratim Das

Further structural diagrams include Composite structure diagram which gives a complete structural view of the system with all its sub-parts and interfaces and so on and the complementary to that is the Component diagram which primarily show you what are the physical aspects of the system so the composite structure if it talks about various interfacing relationship between components the component diagram specifically talked about what components and what static implementation view of this components that you need to describe.

And though we have talking about analysis and design we will find that several of this diagrams actually go beyond that boundary and Deployment diagram is a direct example of that where you not only just talk about the early phase design but you talk about the late phase implementation where you decide, okay.

We finally this system will be deployed over five servers and it has 17 modules you actually decide which module will run in what server whether some module will run in multiple servers whether one module - one server would accommodate multiple modules and so on, that whole description of hardware component and software component mapping is given in terms of the deployment diagram.

Finally, Profile diagram is an auxiliary one which talks about what kind of profile, for example if you are implementing on Java what kind of would you like to use J2EE or you will be using J2ME, if you are using J2ME would you use an Android profile or you will use a IOS profile, so these kind of information's are often shared in terms of the profile diagrams.
(Refer Slide Time: 30:13)



Module Summary

Module 21

Partha Pratim Das

Objectives & Outline

Overview of UML

Why UML?

What is the UML?

What is the UML *not*?

History of UML

UML Diagrams

Diagram Classification

Features of Behavioral Diagrams

Features of Structural Diagrams

Summary

- UML is required to create a standard notation for designing and modeling software systems
- UML majorly contains two types of diagrams, Structure and Behavior to capture a view of the system
- The overall system view is captured by collection of all the required diagrams capturing a certain view of the system

NPTEL MOOCs Object Oriented Design and Analysis Partha Pratim Das

So to summarize we have understood that UML is required to create a standard notation for designing and modelling software systems to liberate ourselves from the vagueness and imprecision of natural language and also not to get trapped in the intricate analytical and expressive power of the programming languages. UML majorly contains two types of diagrams structural and behavioral diagrams.

And we have seen variety of structural and behavioral diagrams overview and the overall system is will be captured in terms of all these different diagrams, so with this overview you will in the next module we will take a brief look into the basic design process design and implementation process of the software to realize how we can start using these diagrams and then we will move into the discussion on specific diagrams.