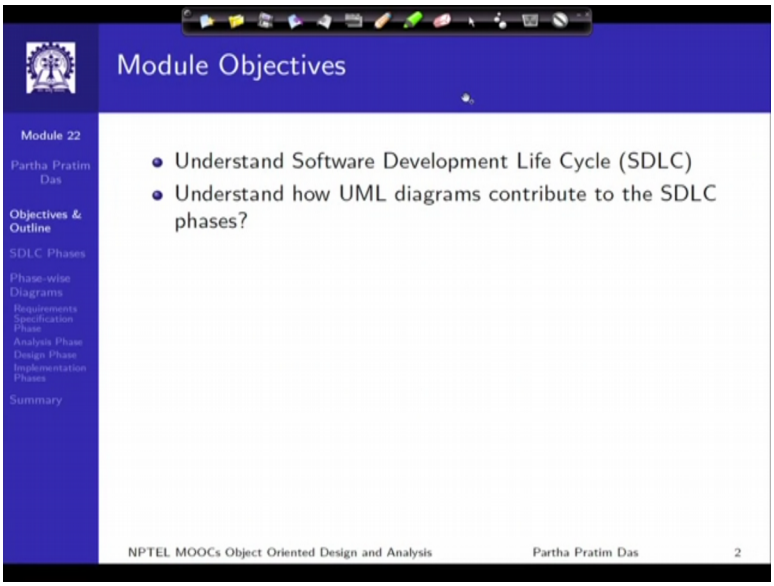
That Object-Oriented Analysis and Design
Prof. Partha Pratim Das
Department of Computer Science and Engineering
Indian Institute of Technology-Kharagpur

Lecture – 34
SDLC Phases and UML Diagrams

Welcome to module 22 of object-oriented analysis and design. In the last module, we discussed and took a brief overview of uml, unified modeling language. We saw that it comprised a collection of diagrams which can be sub classified as structural or the static behavior, static properties or the behavioral diagrams which are dynamic properties and in each of structural and behavioral diagrams there are variety of specific diagrams that are involved.

Now the question certainly is that we want to use uml because we want to bring in better clarity and better expression in the design process. But with so many diagrams around, certainly it is a lot of confusion, fresh confusion as to which diagram should be used when and in what context and so on. So, in this module we will try to answer that question.

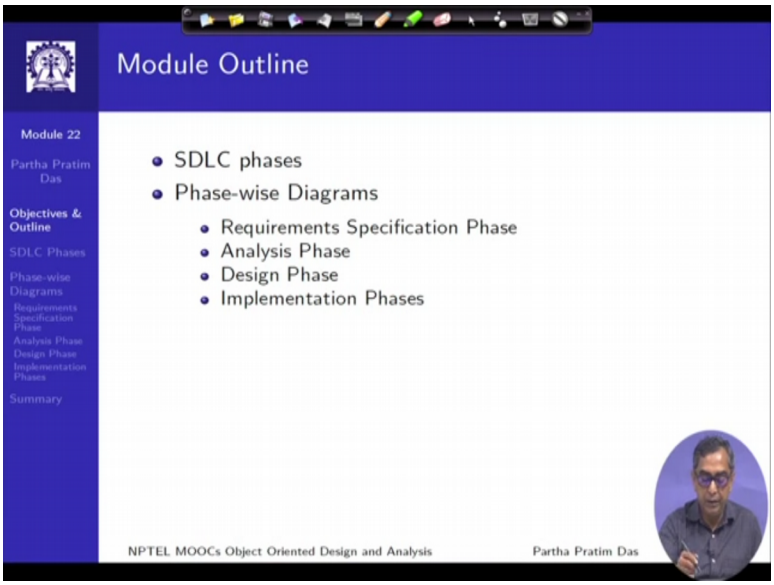
(Refer Slide Time: 01:39)



And to answer that question, we take a look brief look into the software development lifecycle. The whole process of system, analysis and design is a part of n is targeted for the finally the system being implemented as a working software. So, there is a moderately agreed process for software development and it is called a life cycle because starting from the initial development till the end when you finish we see that actually we are at the beginning of the lifecycle again.

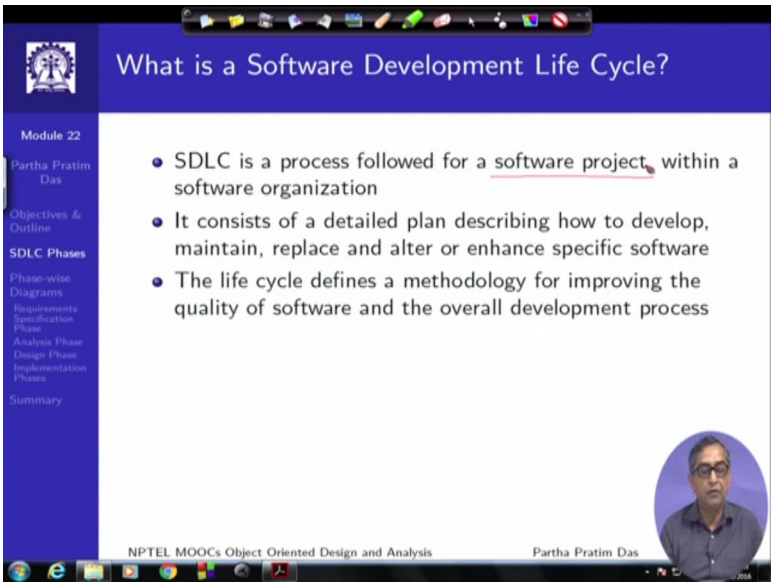
So, we would like to take a brief look into this, some of you, many of you may have had experience with SDLC if you had studied software engineering in some context and once we have taken a look into SDLC, we would try to associate different of the uml diagrams at different phases of the SDLC. So that is the basic objective.

(Refer Slide Time: 02:52)



So, these are this is the outline which will be available to the left of every slide

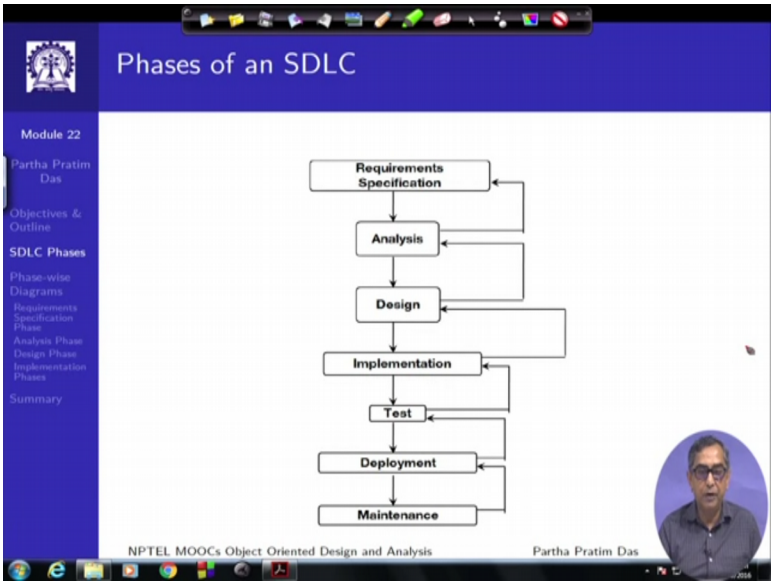
(Refer Slide Time: 02:57)



Now what is software development lifecycle? It is a process followed by a software project or a software organization even for in a software product and it consists of detailed plan describing how to develop, maintain, replace and alter, enhance specific software. So, it's everything that goes on in the time that is software solution is conceived to the time it is deployed to the time it is matured to the time

it is different version created and tot the time that is finally goes out of use is the in under the purview of the software development lifecycle.

(Refer Slide Time: 03:41)



So, it is typically this is not a standard prescription but it is typically software development Lifecyle is divided into a couple of phases. So quickly on these phases, this phase is a requirement specification where actually I should have had a arrow here, where the request or requirement for developing a software has come from the customer from the user. So, who are providing the domain information and we have been discussing a lot about that.

And here the requirement is that you analyze what the customer is trying to say or trying to specify in terms of the system through going through documents, interactions, videos, making notes, images and so on. And generate a very early specification of the system which can be further analyzed for looking to the depth of the what the system involves. So, when you start analyzing it, I would just re relate you I will just refer you to an activity that we did a couple of modules back possibly it was module 20 where we.

We are trying to identify the classes relating to leave in the leave management system and we initially had all leaves being a specialization of a general abstract leave concept and then we did an exercise of finding out what are the different properties, the different leaves satisfy and accordingly we tried started grouping leaves as clubbable, certifiable and so on so forth. And this exercise that we did is necessarily the analysis stage of an SDLC behavior.

So, where you go through all these, refine your captured specification and get into better and better details. Now in this process so this is the this is a forward arrow and in this process, you may require, you may find out that the analysis tells you that well some information is missing, some information some more information is required from the domain, from the customer and so on. If you require such then you need to go back to the earlier phase earlier stage.

And that is the reason you have this reverse arrow and you go back and recapture some additional information in the specification. So, the it it now onwards it goes like that so once I have the analysis output then I am working on the design where I want to more specifically now look at how these concepts can be converted or directly mapped into different software different software details and that too I can do in multiple phases till I am able to generate an output int of the design which will be very detailed design of every system, subsystem module.

If I talk in terms of a my vehicle being C++ has an implementation the out at a output here I am talking about C++ class I am talking about methods, I am talking about specific libraries and so on. So, the design must specify all of this so that the coding of that can start. Now here again you will see that there is a iterative return arrow which is taking me back to analysis for it because the input of design which is output of analysis could have something which is not easy to design or which is not consistent.

And then I will need to go back and do the analysis again or do some more analysis and it is quite possible that I in some cases at least I need to go back to the requirement specification itself. So, you will see that this whole whole diagram is goes something like this, it keeps on doing something and coming back. So, this confirms to the basic principle of iterative refinement that we have had talked about. Now the output of design goes into the implementation phase, this is basically the what we say is coding

and programming where you are using your chosen object-oriented programming language to actually write the programs which realize the design that you have done, again it might mean that you will need to come back to the design at some stage or it might need to go back up totally up to the requirement specification phase. Once the implementation is over, you put it to test which is a very big issue in the in software implementation any of you who have done little bit of study on software engineering will understand this is a very critical stage

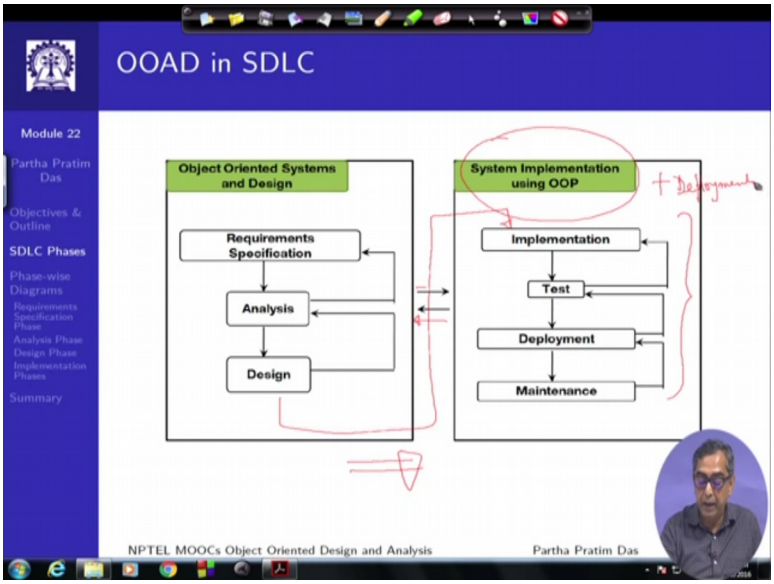
where you are trying to see that whether the 2 main things, whether you have through this process been able to build the correct system that is whether what you have built is what you are required to build given the specification of the customer and the second you try to answer whether you have built the system correctly that is what you have understood in the requirement specification and brought it through all these stages of implementation in the process, you have not made any mistake but you have been able to build it correctly.

So, test primarily is a very extensive process engages about 70 percent of the total effort of software development and tries to answer whether you have built the correct system and whether you have built the system correctly. Again, test anything that fails in the test we say is a bug, you need to go back and make changes in the implementation. Once a system has been tested then the resultant system goes to deployment that is you take it to the customer and put it to customer place and so on.

Let the customer use that and try it out and if it fails then it comes back to test, reimplementation, redesign and so on so forth. And finally, if the deployment succeeds then the software goes into the maintenance phase where you are using it but you may still get bugs at times or it is possible that at a late I mean after using it for a certain period of time, 1 year, 2 years, whatever you need a whole set of new functionalities, you need the software on a completely new platform, you need a different kind of performance and so on.

So, you request to upgrade the software and again from this we will, you go back to deployment or you may directly go back to the requirement specification and say well I have a working software, I have been using it for 2 years now. But now I want such and such and such features included and I start all over again. So, this whole set of phases basically talks about what happens to the if if a software can be considered to be like a human being then whole set of phases will tell you what happens to the life of a software and therefore it is called the software development lifecycle.

(Refer Slide Time: 10:52)

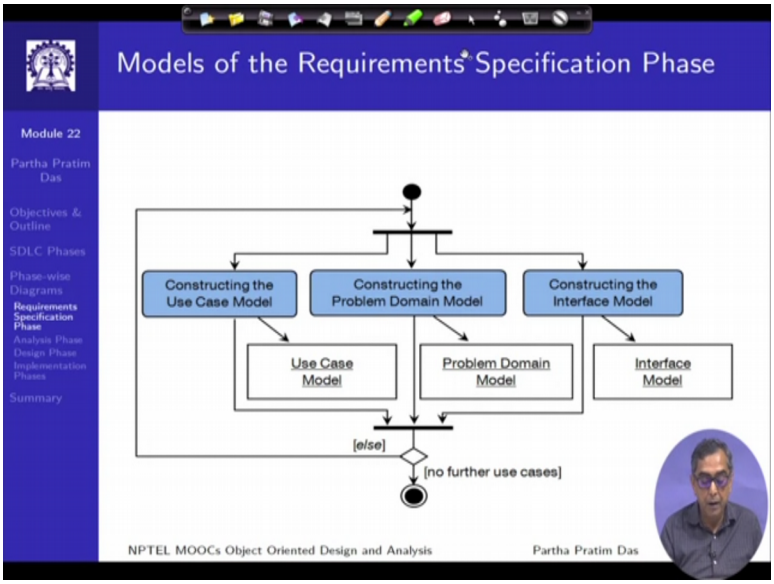


Now if you look into these stages then we can group them into 2 broad parts, the front part which is from requirements to design which I am showing here where the OOAD, what we are studying has a very critical role to play. So, we will see that a whole lot of uml diagrams will be related in this stage. Whereas there is a after the design has been done, then basically after the design has been done, then you go to the implementation as we have seen and then this phase is primarily around the object-oriented programming language.

So, it needs more of in addition to OOAD, it needs a knowledge and understand of the programming systems, the software different software components, the hardware systems and so on. And if there are any issues in the implementation then you will come back to the design again. So, in this part 2, uml all the uml diagrams that we do in the in these 3 parts will go as an input to the system implementation part and in addition I may have a lot of or at least a few of diagram, uml diagrams which will take part in the details of this system.

So that's the basic role of OOAD in the SDLC and we will keep that in mind to identify the different diagrams.

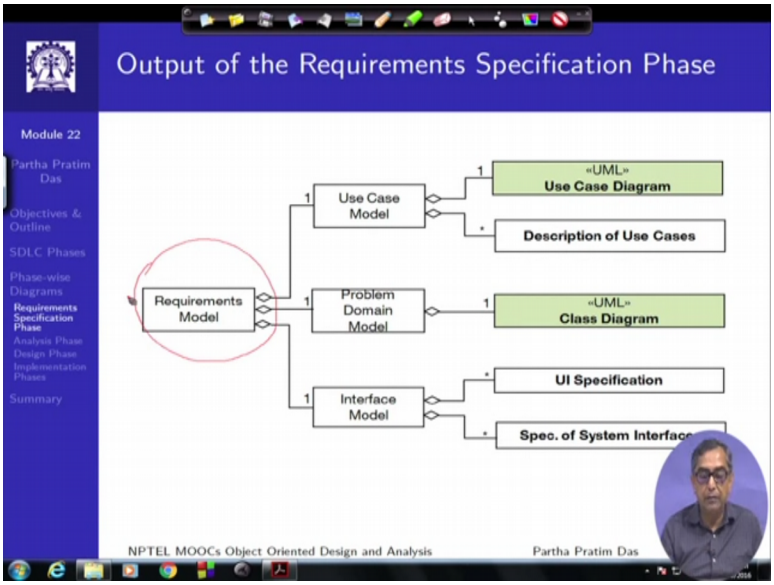
(Refer Slide Time: 12:19)



So, starting from the first phase which is the requirements phase that it typically uses 3 different kinds of models or diagrams. So first you need to construct the use case model. So, you use the use case diagram, you need to construct the problem domain model which is actually trying to capture what are the basic properties, key abstractions, relationships of the problem domain are. So, use problem domain models and you need to have an interface.

So, this interface I mean the moment we say interface it it kind of a suggests as a graphical user interface GUI but it may not necessarily be that. GUI is also included but it could be all other kinds of interfaces including keyboard based interface different kinds of voice interface, auditory interface and even program interfaces which come in the interface model.

(Refer Slide Time: 13:22)

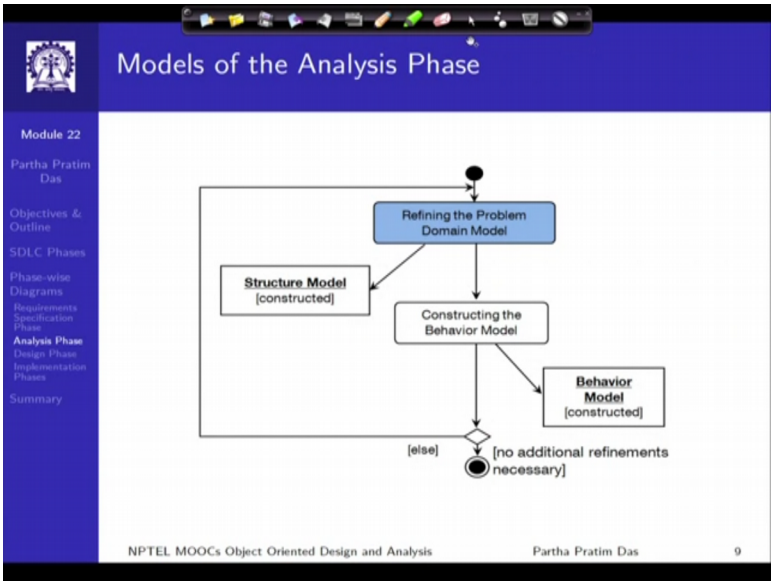


So, when we do the requirements phase, requirement specification phase, then we expect that we will come out with a set of models which as a whole we will refer to as requirements model and this notation we have used it to show aggregation at at a certain stage earlier this show that basically this is a union of or comprise the use case model, the problem domain model and the interface model. So that same uml notation is used to describe uml here.

So, the requirement models have these 3 types of models as we have seen and in terms of the uml diagram, this is what the use case model will do. So, when we say we need to identify the use case model, how the system can be used by whom to do what, we capture that in the use case diagram. So that will have to be the output of the requirement specification. Similarly, class diagram will identify the basic objects and classes and is a part of the problem domain diagram that problem domain model that we need to create.

In terms of interface model yet uml does not have very strong diagrams, so UI specification and system interfaces and so on are still often done in a little bit unstructured way, little bit with the use of you know natural language and hand drawn drawings and so on. But then in these 2, the use case and class will be the core components of what you expect to generate after the requirement specification

(Refer Slide Time: 15:08)

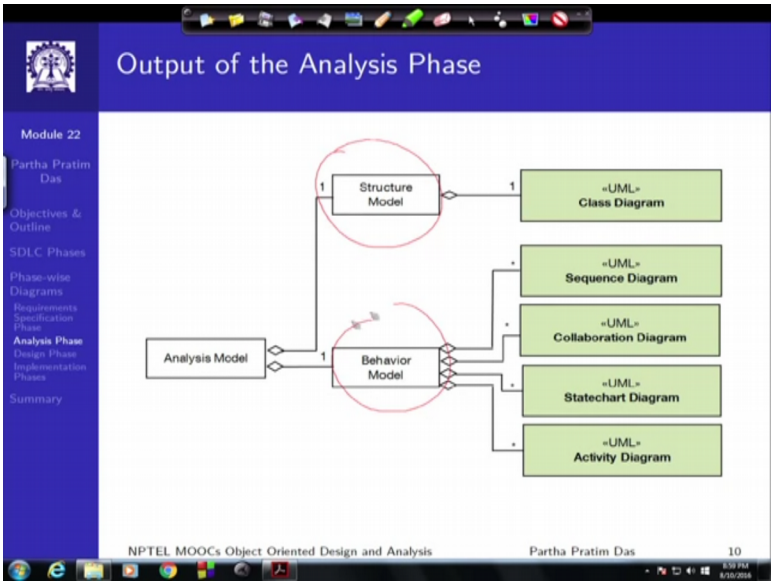


So naturally these will now go into the analysis phase and in the analysis phase when the main structure is analysis and to define the problem domain model again remind yourself of our exercise of refining the way we had modeled the hierarchy of leaves. So that is the activity that you are doing here. Naturally what you expect to generate our structure models which are constructed and you start

constructing the behavioral model.

In terms of the requirement phase output possibly you can still you can still have certain behavior description in the class but is primarily a structural model but when you are completing analysis, you are expecting more and more of the beh behavior to be extracted and behavioral models to be explicitly constructed. So that is your dealing with uml diagrams in the analysis phase and as a output of that, this is what you expect to generate, a set of structure models and a set of behavioral models.

(Refer Slide Time: 16:13)



The structure model is again class. So, you can note you saw this in the requirements phase output also. So, you again say this it does not mean that the fact that you are is using the same diagram does not mean that it is necessarily exactly the same diagram. It is a class diagram that you are using but the set of diagrams you did at the end of requirement specification through analysis that has got refined and you may have a differently organized class diagram, you may have a lot more of details in the class diagram.

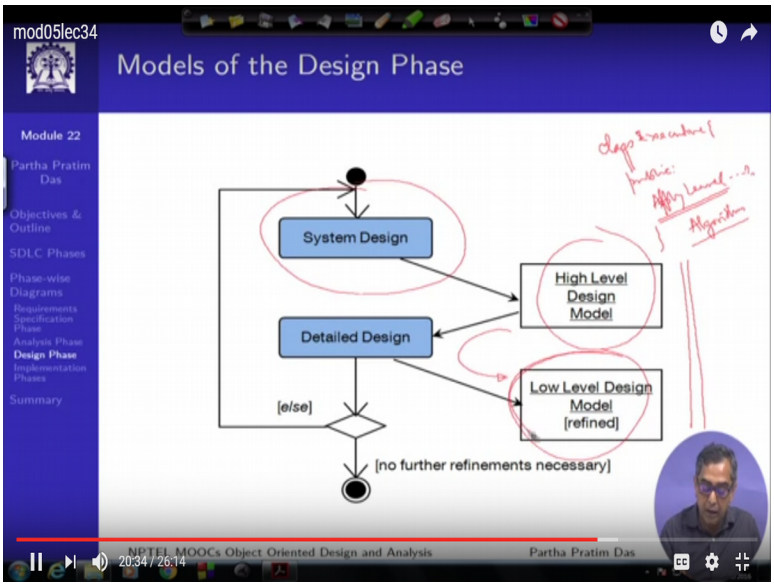
Now again the example is at the output of requirements phase you just have one leave class and all those leave are considered to be specialization of that and at the end of the analysis phase, you have the same class diagram, now refine in terms of a complex hybrid hierarchy. The behavioral model includes these 4 which I briefly touched about in terms of sequence diagram. The communication or collaboration diagram, collaboration diagram I have,

The term is mentioned here because I I mentioned that will switch will keep on switching between the

stand the current standard and the earlier one-point x standard. So, one-point x standard used to whatever one-point x standard used to call as a collaboration diagram from the 2-point x, this is now called the communication diagram. So, you can this is same as the communication diagram, not a different diagram. the state chart diagram to show the different states and the activities.

So, these are the diagrams which have to get created through the analysis process and each one of them will capture a different aspect of the system.

(Refer Slide Time: 18:02)



Naturally the output of the analysis phase goes into the design where the main activity is the system design that is now it is expected that you have been able to understand the specifications, understand the problem domain, the requirements very thoroughly and you have been able to represent them in terms of this whole lot of different uml diagrams and now keeping all these in the in your mind, you are now trying to design how actually we will go about organizing the code,

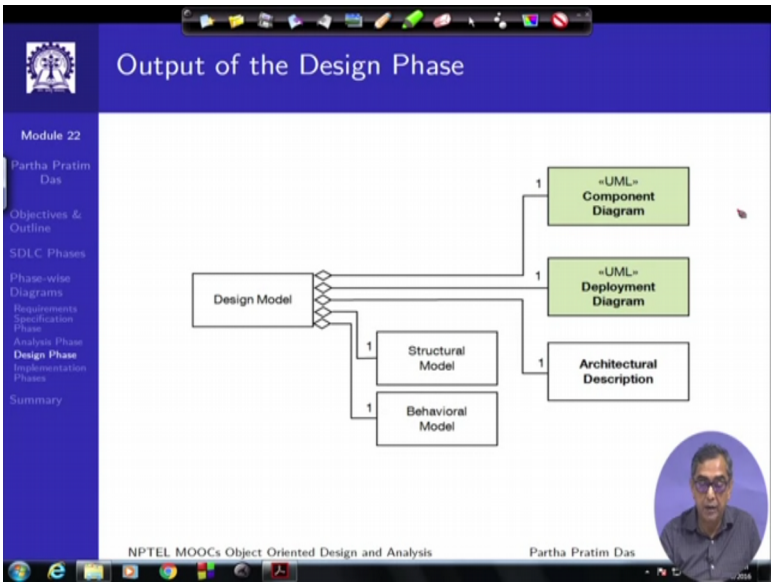
what should be the different code component, what should be the actual classes that you write in the oop and so on that is what you mentioned about system design and as you can see that its typically done in 2 stages. The first stage is you do a high level design model. For example, if in the leave management system, we decide that we will have we have (()) (19:00), we know that we have class like employee, executive, lead and manager.

Then at this stage of the system design stage, for if C++ is my vehicle since I have a class executive and so I I write it down and I say that this will have different methods like apply leave kind of and so

on. So that is I I work out a basic high-level design. Then I want to again take it through another design phase which is called the detailed design where I will refine on this high-level design. So here possibly I was happy saying that I have apply leave, here I would like to know the algorithm that will be used in applying for leave.

I will glad to know that do I have some idea about at least what are the data members that I should have. These data members are nothing but attributes of the class but if I say that the attribute of the start date has a for the leave, it has a or rather we are talking about executive so it has a joining date of the executive, it has the age of the executive and so on. We really need to detail out as to what kind of type the every data member should be, what should be the size and so on. And those are the tasks at the low level design phase.

(Refer Slide Time: 20:58)



So in in short you will find these terms very widely used. this is called HLD high level design and this is called LLD and this whole process of design will split out again a set of different uml diagrams. So now please keep this in mind that is like go over from one phase to the other, all outputs of the earlier phases still exist and are still valid unless they have been rejected and removed. So, we already have all those use case, class diagram, different behavioral diagrams of state chart sequence

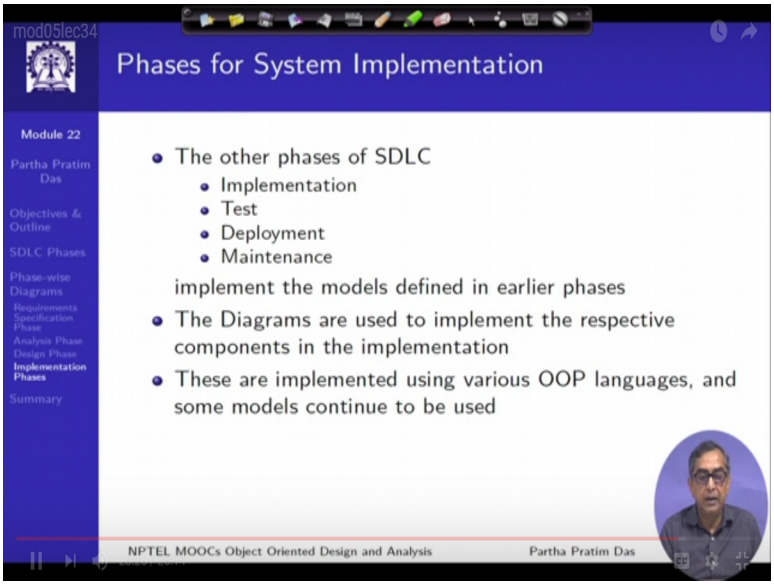
and all that but now in addition we have been able to create these kinds of diagrams like component diagram. So component diagram will tell you am saying leave management system it will tell me that I have an employee component which implements variety of employee related classes, employee executive, lead, I have a leave component which certainly talks about different kinds of leave classes, I

have a joining component which talks about how does not employee start existing in the system and so on.

So, I talk about all those and I also outline the basic diagrams in terms of what is going to be my target system. What kind of hardware do I have and how do I lay out these components in the target. So, what is my deployment plan that we will come around with. Now along with that I may have other structural models, behavioral models that I have already got from the earlier phases and in addition there could be some more details, now as you can see that here I am calling this as architectural description.

If you could refer back to the last module, module 21 and look at the hierarchy of uml 2.5 diagrams you will find that in terms of deployment, people are often now talking about say network architecture diagram. So, in in in may be 5 years back we would have been just happy to talk about a general you know English like description or diagrams defining what is architecture of the system, how are different servers and front ends and network cables are connected. But now we may need to slowly work towards formalizing that in terms of better diagrams as well.

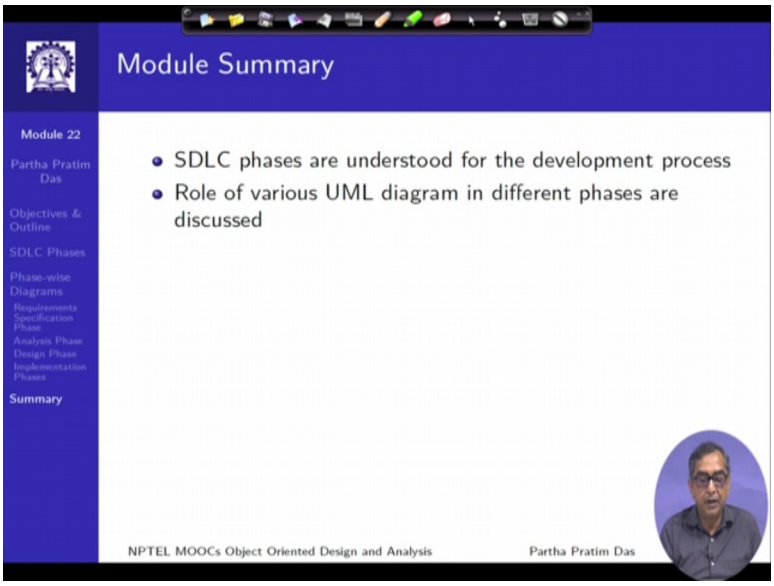
(Refer Slide Time: 23:18)



So, this is the kind of diagrams in the design phase and that is the those are the 3 stages, phases of SDLC requirements, analysis and design where uml's have a big role in terms of getting formulated. Then in the later phases of implementation, test, deployment, maintenance, their main role is that these diagrams are then used and the implementation is designed. So, in this other phases, it is typically the use of these models but certainly in that process also we might generate some more details in some of these models.

We may need to go back to the design or analysis phase or even requirement specification phase to refine the models and work through that.

(Refer Slide Time: 24:11)



So, to summarize in this module, we have tried to take a brief journey around the commonly accepted software development lifecycle process. I have talked about the major phases of this life cycle and how the uml diagrams will be used in different phases or will get generated in different phases. So, from the next module onwards when we specifically talk about the details of every diagram, we start learning about what their primitives are, how they should be formed and so on.

Then we will be able to relate to which particular stage of SDLC software lifecycle we exist. before I close, I would also like to put a disclaimer in terms of the SDLC because the kind of approach that I have shown or discussed in the SDLC here in there most classical and the flat approach, there are several different SDLC variations that exist today starting from different agile methods as print method, test driven methods and so on.

So even if you are actually using a different SDLC flow you will find that maybe the checks and balances of what phase you do, when we complete a phase and how do you repeat a phase or how do you refine a phase, those details will vary but you will find that whatever methodology you may be following, these stages of requirement specification analysis design all these kinds of activities would be there in the SDLC that you are following.

So, once you quickly identify that, you will be able to also relate to which particular uml diagram you should be using for doing what kind of task at different stages.