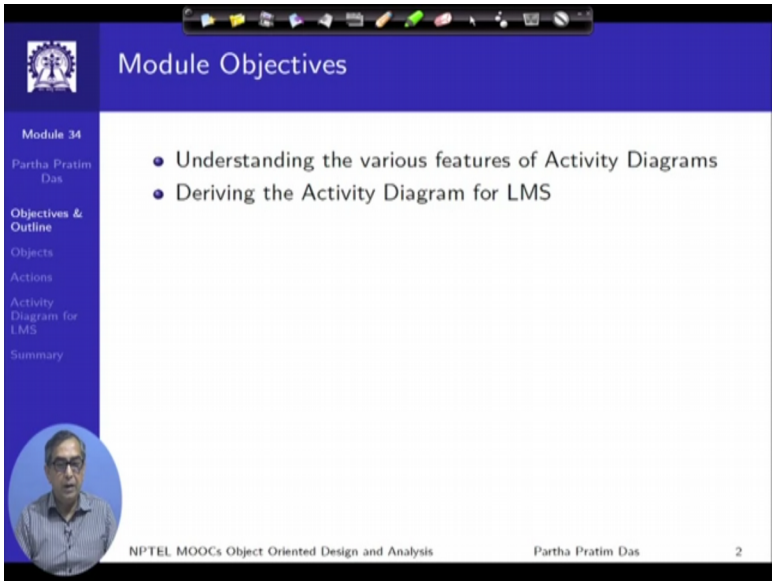
Object-Oriented Analysis and Design
Prof. Partha Pratim Das
Department of Computer Science and Engineering
Indian Institute of Technology-Kharagpur

Lecture – 46
Activity Diagrams: Part III

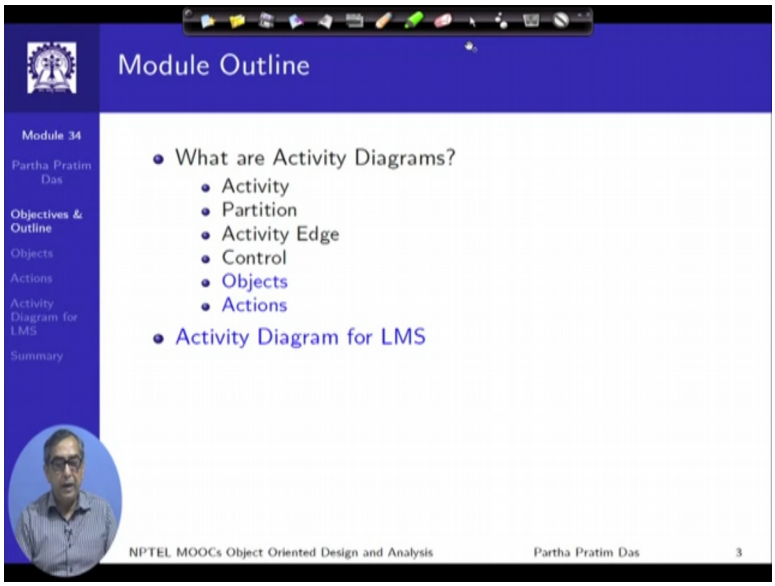
Welcome to module 34 of object-oriented analysis and design. We have been discussing about activity diagrams in uml in the last 2 modules.

(Refer Slide Time: 00:34)



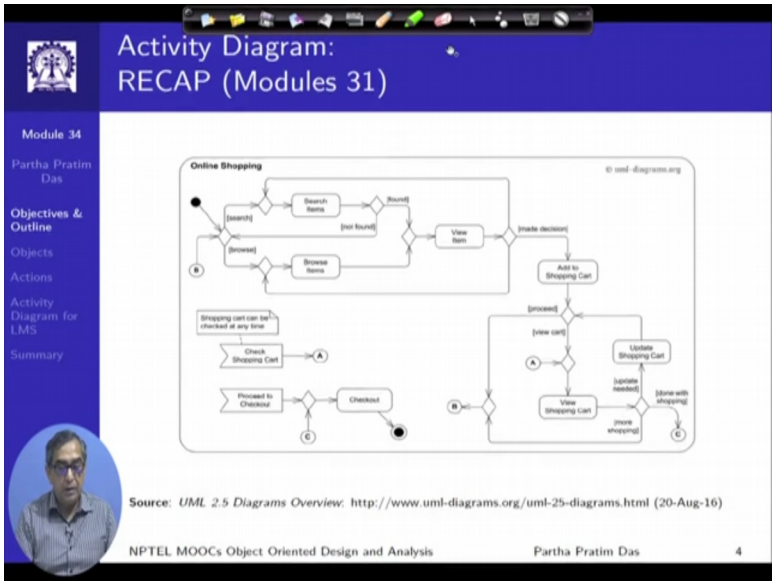
We have talked about several of the features of activity diagrams including the activity themselves the partitions, wedges particularly object wedges and then in depth about control nodes and how what role the control nodes play and we have seen some examples.

(Refer Slide Time: 00:55)



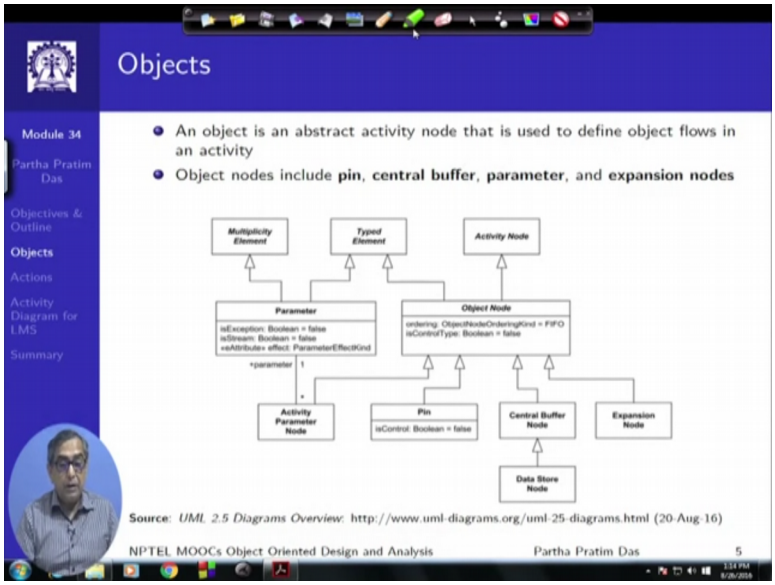
In this module will continue further and we will specifically take look into objects and actions and then we will take a brief look into the leave management system in terms of the activity diagram.

(Refer Slide Time: 01:11)



So, this is one diagram you can you can recap in terms of what we have studied earlier.

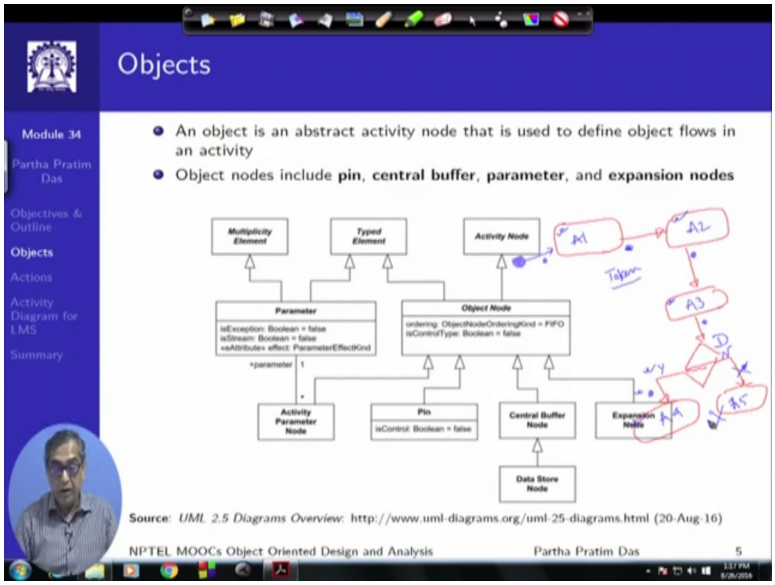
(Refer Slide Time: 01:21)



Now the next concept that we have in terms of activity diagrams here are objects. The object is an abstract activity node that is used to define object flow in an activity. See the way you can you can conceptualize if you let me just go one step back in terms of the conceptual model of activities and their diagrams, say suppose there are 2 activities here and there is an edge here, now there could be that's a there is a there is another activity here having edge here,

There is a decision here, there is a decision here, giving you different activities on this. Now as an active this is a representation of the flow how do you really understand the dynamics of the flow; how do you really understand that all the flow is going. So, the way you conceptualize is you think about certain abstract objects, these are typically called tokens, that move from one activity to other. So initially when when if we represent we have the initial starting mode, the token is here.

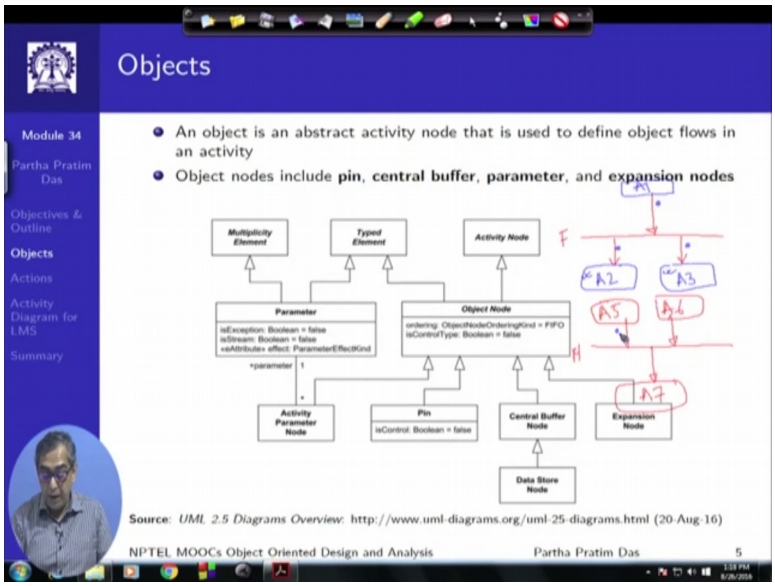
(Refer Slide Time: 03:05)



Then this activity happens say activity a1 then the token has reached at this point. When it reaches this point the token gets transferred to the edge and activity a2 starts. Once this is done, the token comes here, so it is as if the actual dynamics is as if you are tracing here tokens then you have a3, so when does an activity start? When all its inputs token have been received. This is these are simple flow so they have only one token so as soon as that token comes, this activity happens and once it completes, token is goes to the out.

So, when you have a decision node naturally after this there is one token.

(Refer Slide Time: 03:47)



So after this decision node either the token goes here or the token can go here, it cannot go at both. So, if this is yes so this is no and if yes is taken, then this token comes, this token is not there. So, this token has come, the activity a4 will get fired but activity a5 cannot get fired. That's the basic concept of activity flows in in general. So, you can you can understand that very well in terms of a

fork situation that we have one flow coming and let it let me write clearly.

So, one flow coming here as 4 and you have 2 flows going out. So, what happens is is in terms of this you had some activity here, you have some activity here as well as some activity here. Now when this activity a1 has got completed naturally a token is placed on the output edge that goes to the phone. What the phone does? Phone basically duplicates this token on both the output nodes so that both these activities have a token in the input and they will fire, they will start fire.

Similarly, when we have a merge the reversing happens. Have a merge so I have 1 control coming in here, 1 control coming in here, 1 control going out, there is an activity here, there is an activity say say A5, A6 just arbitrary names and another activity A7 here. So, in merge what happens is whenever this activity A5 say will get over, it will place a token on this. But merge will not transfer that token here, it will wait for the other token also to come.

So, the merge node basically waits for all tokens to come, merges them together, places the token here and only then this will go on. So, this is the same logic I have said but the only difference am asking you to conceptualize if you think of a token traveling along the edges of the activity diagram with the further assumption that if I have an activity then naturally it may have multiple edges coming in and the activity will fire.

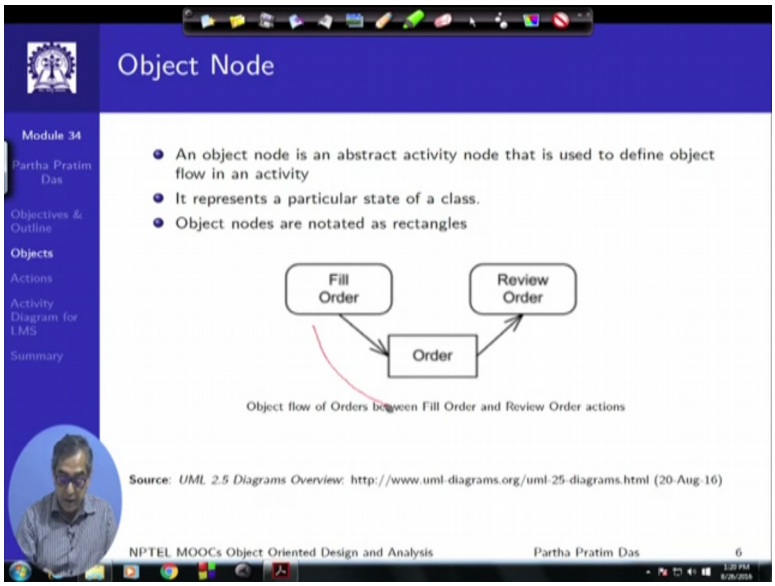
So, if multiple edges have to come in you either have a merge node or a emm or or kind of join node to represent that but only when the token is available on all of them, will this activity fire and once it fires it will place the token on the output. If you think about this then that becomes much easier to understand. So, whenever we are working with the activity diagrams and 2 activities A1, A2 we put an edge between them, we do not merge this token, this is implicit.

This is the basic behavior so that A1 when it will complete it will generate token that goes to the edge and A2 has a token and starts working and when A2 gets over, the token comes to this. Now specifically what objects the role that objects play is if at places you want that beyond this generic behavior of tokens to manage the flow, if you want some more packaged information to go along with that, you put it in terms of an object.

So, it's a same token but its this here the token is not an anonymous one without which is just to control the flow but it will also carry some information. And it could put the object in variety of different shapes and sizes and different purposes to satisfy different requirements. So, it's best to

just to have a quick look into those.

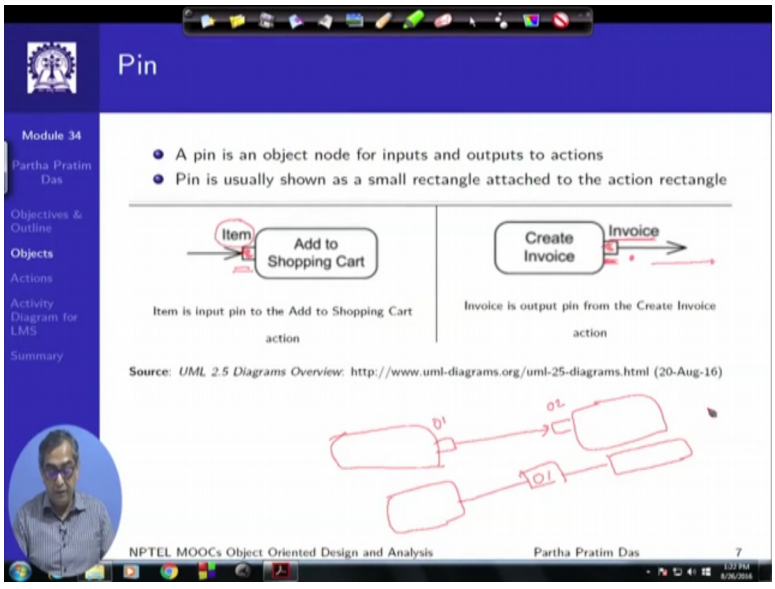
(Refer Slide Time: 07:43)



For example, we have already seen this. So here if you just think in terms of the simple model it is an edge of from fill order to review order which will carry a token. Now we have refined it saying that I have order object here. So, fill order which when it generates the token it actually generates a qualified order object and that order object goes in here. So, on the edge you see the order happening and this.

So basically, these objects will always happen on the edges because they are basically refinements of the token that has to go through that edges.

(Refer Slide Time: 08:18)



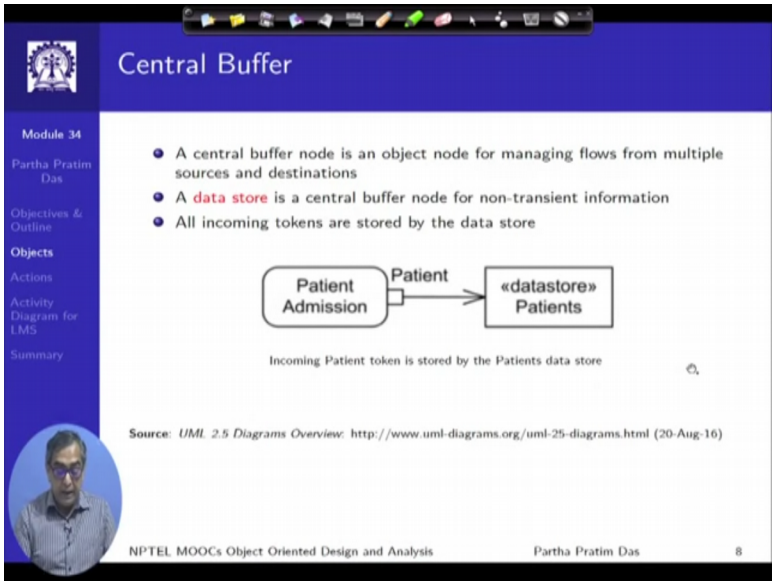
So, another concept is a pin object where you show that well this is an input pin, this is an output pin. So, this shows that when the token arrives to add to shopping cart, you are showing that it's just not an anonymous token you expect and the token in terms of an object. So, it should have other

properties also and you are specifying what object you expect here. That's the item object. Here it's an output pin, it's an out-pin token, so when that create invoice completes the token is to be placed here.

You say the token is not a simple one, that is an object and that that object is identified as an invoice. So, wherever this goes wherever this goes it will get that object along with that token. So that's a you can touch that the objects to edges or you can attach them to the pins so if if I if I connect them end to end so then this will look something like this. .so, so there is an object here, the object here and so on. So, this is how the pin objects will work, the they look like very similar semantics.

Because I could instead of going this I could have just done an O1 object and could have sent that to the next activity, the reason you you bind it to the pin because it you may have different kinds of flows. For example, if if you have the same 1 object generated which goes to the multiple of the different activities then it is easier to show them in terms of pin or if if you have that different activities can generate an object of the same type which we expect at the input of an activity, then it's easier to show it as a pin.

Also pin provides for certain transformational objects in the process. Will show an example.
(Refer Slide Time: 10:11)

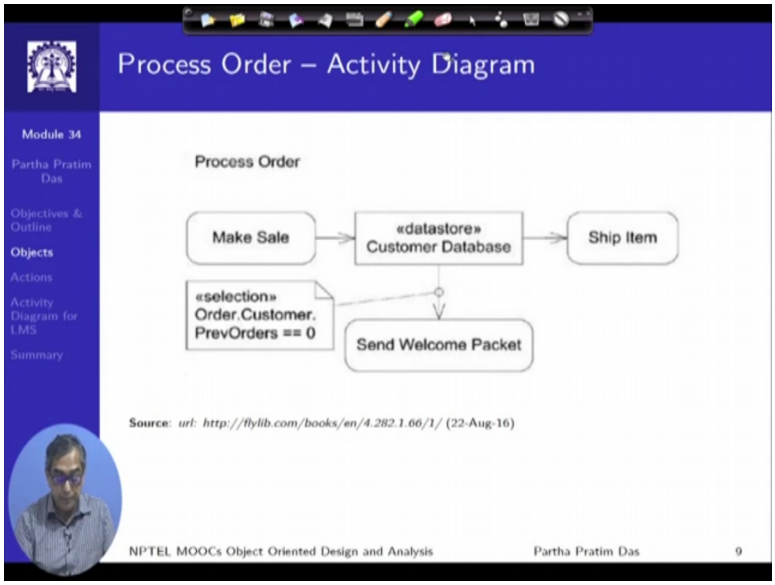


There are cases of what is known as a data store, often you will need a data store to maintain the collection of objects and so when you talk about the data store. So, data store is again shown in terms of a keyword. So, there could be several data stores which you could relate would later on become possibly tables or in some cases databases in your central database system. So, this data

store we are saying that the patient admission this activity ends, the token is generated, it is an outbound object patient and that's delivered to this object datastore.

So here kind of you do not have another activity following it, so this is kind of an object which just works as a repository. So, objects could be used for such purposes as well.

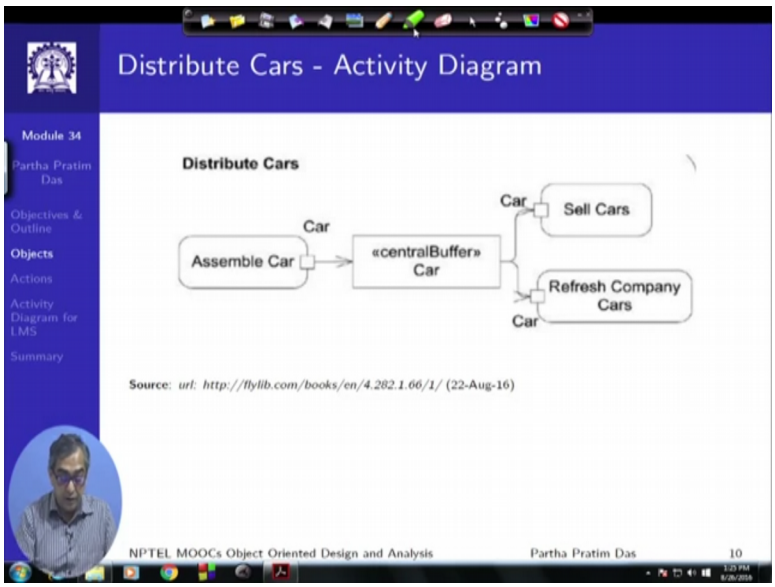
(Refer Slide Time: 10:58)



So, with that you can have variety of these there is an activity here which sends an object which on completion sends a token to the data store which is a customer database. So, making a cell that basically says that you have updated the customer database and then from this the activity, next activity ship item which means that based on this update the shipping of the item will happen and the the outcome the transformational outcome in this is send a welcome packet.

So, where what you are saying that when this object goes to the edge then you send an outcome welcome packet based on certain selection criteria. So, objects are very objects as you know qualifiers of the edges are an important concept in activities and can be used in such multiple ways.

(Refer Slide Time: 11:55)

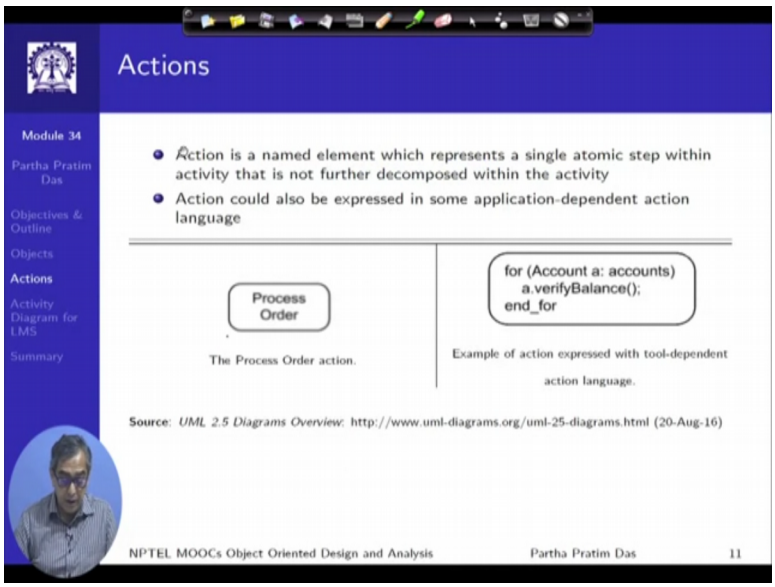


This is one another example of distributing a car, another activity. So, you assemble a car, this is an activity, on your out pin you have a car, because you have assembled it. That goes to the central buffer so that's kind of the garage where the assembled cars exist and from there you could either sell or you refresh the company cars. These are the different activities that can that can happen. So, this basically is is a is a multi-way activity that is happening and each one of them get a car from this store.

So here you can see that this is being a central buffer, this is different from if I have just said that from here to here I have a car. The difference being that if I had just shown an activity assembled car and sell car connected with an edge car so that you assemble the car and sell car that would mean that you assemble the car and that particular car you go send for selling or you assemble the car and that particular car you send for the refresh company which is not the case.

So, the central buffer gives you a different semantics where you assemble car and you accumulate them here which is some separate process and here you have from that central repository you pick up cars and sell them and refresh them and so on. So, this is how you can objects can be used to add more semantics into the activity diagram.

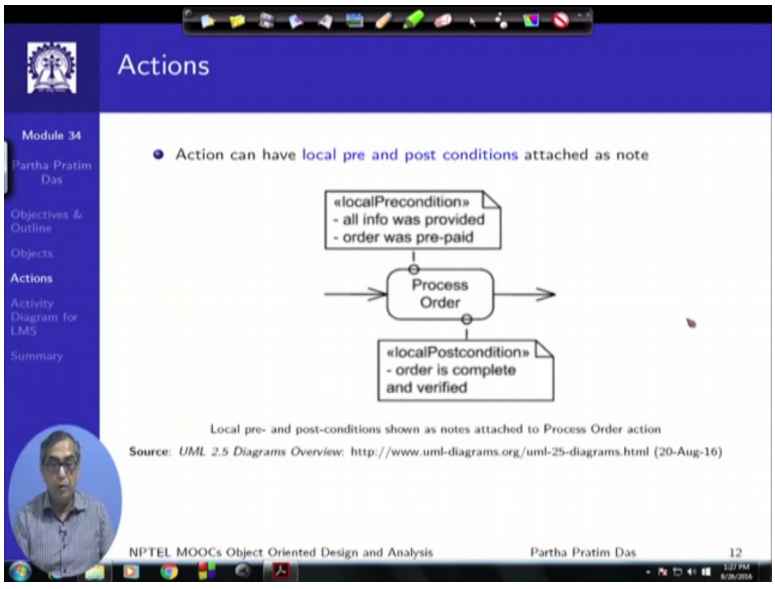
(Refer Slide Time: 13:18)



Then you have other than objects you have actions am sorry actions, they are they are also named elements. So, the action is a that the process orders action. So what action mean is basically it means there could be you know these are activities are algorithms at a certain level but you could you may want to actually have a more direct algorithm or programmatic representation of that. So, you could also define actions which are expressed in certain programming language.

So, actions not necessarily are generic in nature you could have actions in and uml activity diagram just specific to the uml tool you are using so that you can give more and more details, for example you could have a c code attached as an action saying describing that how it will be realized and so on. So, actions give further details and low-level details in terms of that.

(Refer Slide Time: 14:23)

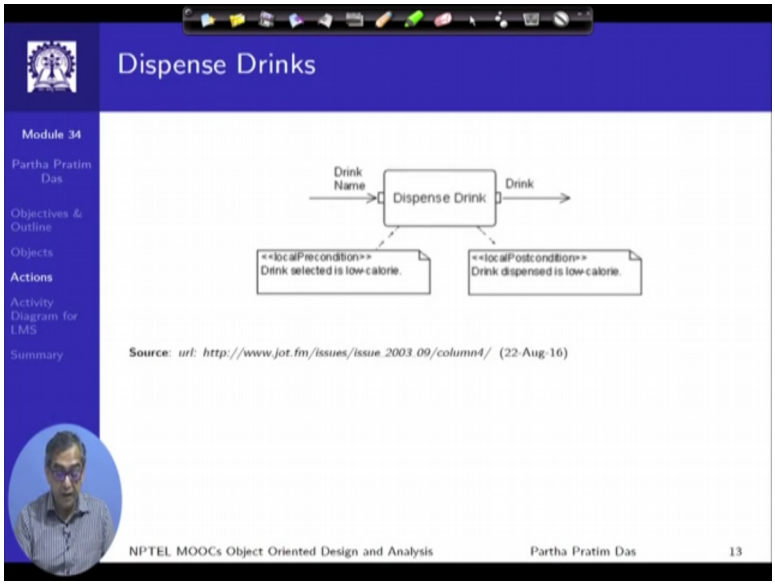


And certainly, since you want to model the overall activity. so, actions could have local precondition and local post condition which say that if this action has to happen then what are the conditions has to be satisfied for this action to take place. That's the precondition and what will

happen after that. So, if the pre-condition here of processing order is all information was provided and the order has been prepaid, the payment has been made, only then the order can be processed.

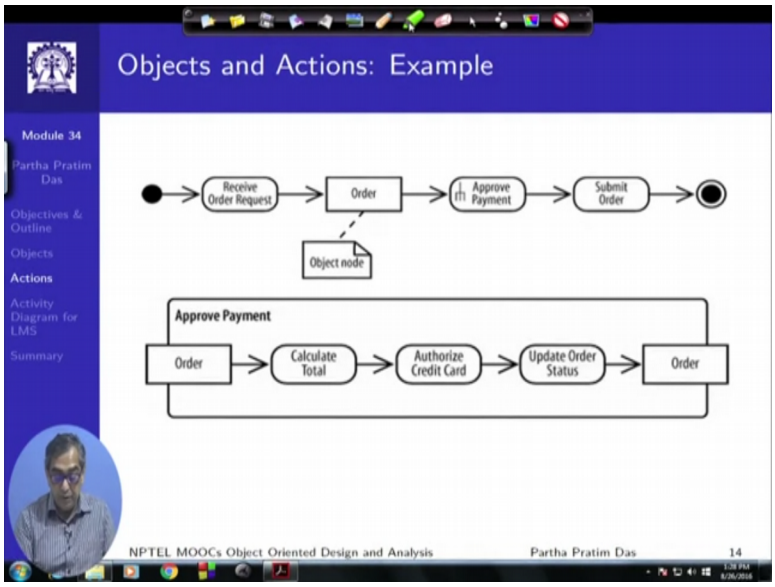
So that's the precondition to this and the post conditions certainly is the order is complete and verified. So, actions take place in this way.

(Refer Slide Time: 15:00)



So, this is another just showing dispensing of a drink. So, the precondition the action of dispensing drink as a pre-condition that whatever the user or the customer has selected that is drink is low calorie say and post condition has to make sure that the dispensed drink is low calorie, again see pins here the input and output pin. The input pin here that gives you the name of that drink so that the dispensed drink activity can actually pick up that drink so the output pin gives you another object which is a drink object itself.

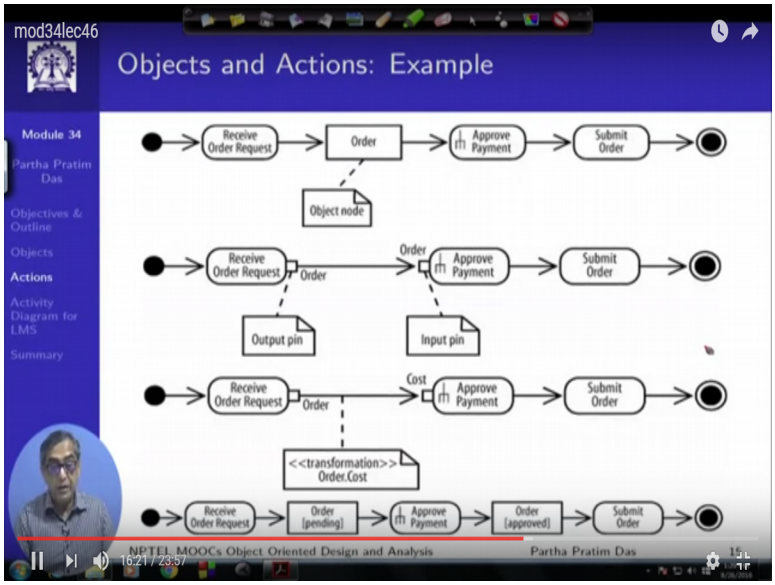
(Refer Slide Time: 15:40)



This is another example of processing order so you you receive your order request and you send it

for approval of payment, you are showing that order object in the middle which is the object node and then this goes to the submit order in this activity. You can see this notation of sub activity here, so approve payment is another detailed one which takes the order, calculates the total, authorize the credit card, update order status and finally makes the finally updates the whole thing in the in the order itself.

(Refer Slide Time: 16:20)



So, if we look into this flow we can see that in the same flow we can look at the object in multiple different ways this is the first one that we saw, the receive order request and send it for approval of payment you have an order object here. We can show the similar information this is just am just showing different variance of the same diagram but with certain sudden changes in the semantics. So, this I can also show in terms of a pin, this is an output pin in the receive order request which actually has generated an order on the output pin

And the approve payment in the input pin gets this order on the input pin and rest is same. Once we change from this representation to this representation, then I can make further changes here, I say that okay what the approve payments needs to know really not the order, what it needs to know is the amount of money that needs to be collected. So, when the output pin object could be an order, the input pin object is actually a cost that am interested in.

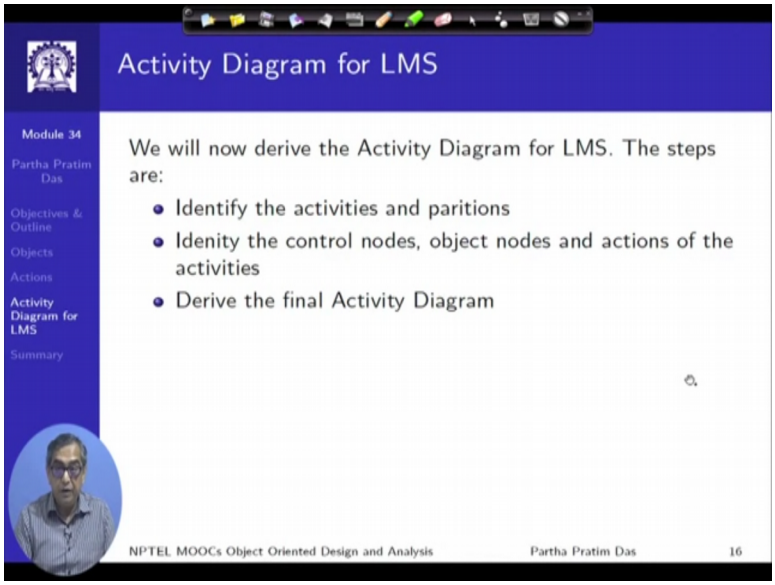
And so, I can tra have different kinds of objects at the 2 ends of the activity edge if I use pin and I can bind some transformational algorithm for converting one kind of an object into another kind of object. This is represented in terms of the transformation keyword and which here the it is simply do doing a selection which is taking the order object is just checking out the cost component of it. So, it is order dot cost which gives in here.

There could be several detailed algorithms, there could be an action involved here and then you proceed with this whole thing and at a different level I can also represent this in terms of this request that are receive payment the object order object goes to the approve payment but am not getting into these details. All that I know in this object is in a state which is un appending because this order is gone for approval.

So, this order is an order object but this is a pending status whereas once the approval has been done in in the earlier cases if you look into these cases we dint show we dint show any object flowing, it was the default token. But here we show that as if an object is this order object is flowing from the approve payment to the submit order, the same activities. Only thing that has changed is the status or the state of this order object. So, this is a modeling which is based on the state change information, later on when will talk about the state charts.

Will see how different such state information of object could be combined in different places in uml. So here I just wanted to show you the the power of you know finally modeling the activities not only in terms of control flow which is what we have been doing in the last 2 modules in terms of activities, partitions, edges and particularly the control flow diagrams. But here what I illustrate is on top of this you can make a very strong object flow on the activity diagram and the token that move can carry whole objects with them defining a complete semantics for the diagram.

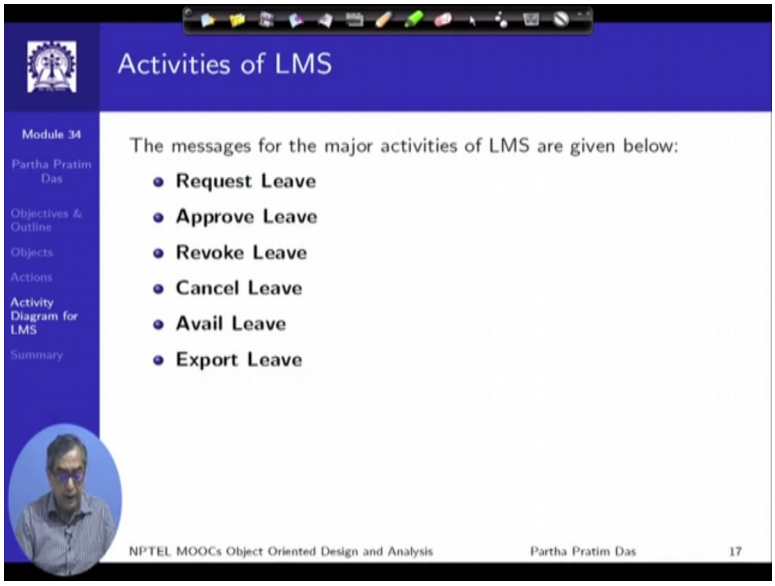
(Refer Slide Time: 19:51)



So that was basic overview of these activity diagram. So, if you look at lms are running example then we need to identify activities and partitions, the control nodes, the object nodes, actions for activities and derive for activity diagram. Of course, what I show you here is very elementary form

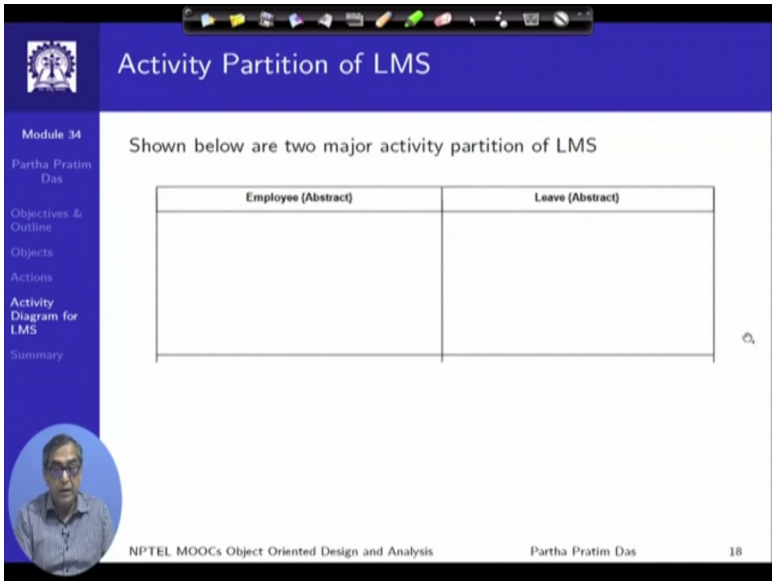
of the activity diagram. I will expect you to work this out at home so this is just to show you the outline.

(Refer Slide Time: 20:19)



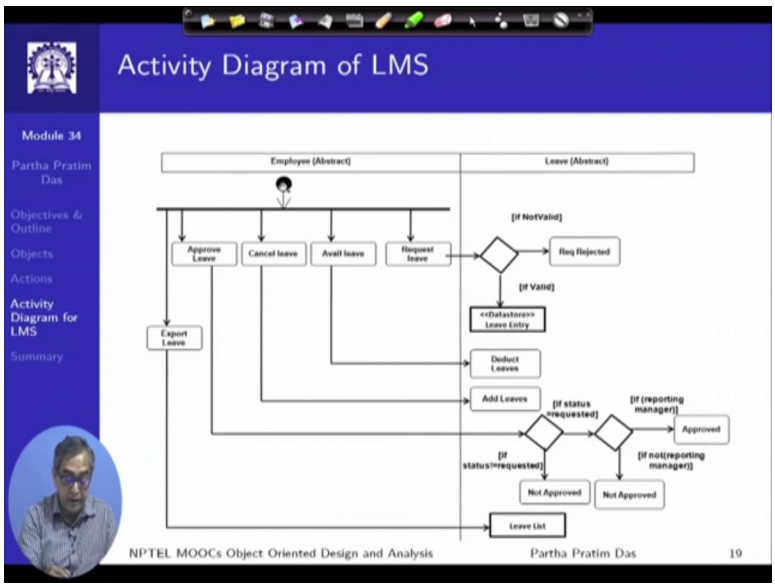
So, the different activities that we have here would clearly get in in case of lms, they will clearly ba be identified from the use cases. So, these are the different use cases that we can say will become major activities in the activity diagram. Managing leave in multiple ways, request, approve, revoke, cancel, avail, export and so on. So that's simple based on the use case diagram that we have met.

(Refer Slide Time: 20:48)



In terms of partitions in swim lines at least 2 swim lines we can immediately think of based on the 2 major agents that we have that is the employee, there is I will be activities the employee has to do and there will be activities on the leave abstraction that needs to be performed.

(Refer Slide Time: 21:13)



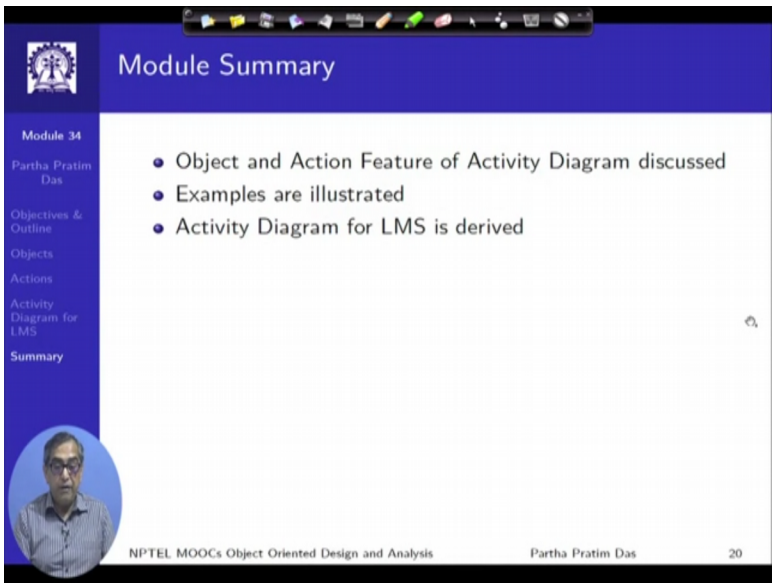
So, if you draw it on this streamline then we can say that well we start here. .and when we start then it is possible that different employees will take different actions, different activities of requesting leave or approving leave or availing leave and so on. They all start with a fork because they can start concurrently anywhere and then it goes on with the different activities and if you request for leave, it process over to the leave swim line and depending on whether it is valid or not valid, you have different decisions, outcomes of rejected leave or leave entry.

So, this is the central buffer store object where if it is valid then the leave is entered in the store then avail leave then it will be detected certainly there will have to be further if you cancel leave, there will have to be further activities that will come in. if you approve then you will need a decision to check if the leave is indeed requested. Only then you can go for approval, if it is not then it is not approved, if it is requested then you need to check if the person trying to approve the leave is actually the reporting manager.

So based on that the leave can be approved or otherwise the leave will be, of course these are there are further and further details and what is significantly shown missing in this diagram is we just have the fork that we do not have to join, that is finally it is not combining into some common place, so we need to figure out how to do those and complete it in terms of complete activity diagram of lms, when we are at the end we will when you provide you the uml solution for lms, will give you a more detailed diagram.

But I will expect that meanwhile you work on this and try to build up the activity diagrams for lms as well as the ams and may be some of the other systems that interested.

(Refer Slide Time: 23:16)



So, to summarize we have discussed about specifically about objects and actions in activity diagrams and overall this concludes the three modules through which we were we have exposed you to different aspects of the activity diagram, how to model them, how to use that and we have taken up several examples to illustrate the different features and the use of activity diagram and have given a brief over the activity diagram of the leave management system.