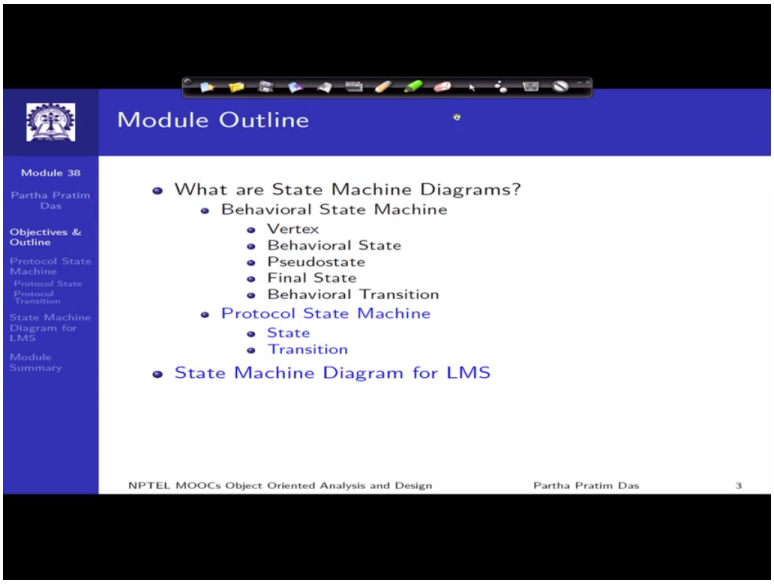
Object-Oriented Analysis and Design
Prof. Partha Pratim Das
Department of Computer Science and Engineering
Indian Institute of Technology-Kharagpur

Lecture – 50
State Machine Diagrams: Part III

Welcome to module 38 of object-oriented analysis and design. From the last two modules we have been discussing about state machine diagrams. These are uml behavioral diagrams which are very important in refining and capturing the discrete behavior of a system in terms of a set of finite states and a state here is a at a higher level of abstraction than the state as we refer to them in terms of a finite state machine or a regular automata.

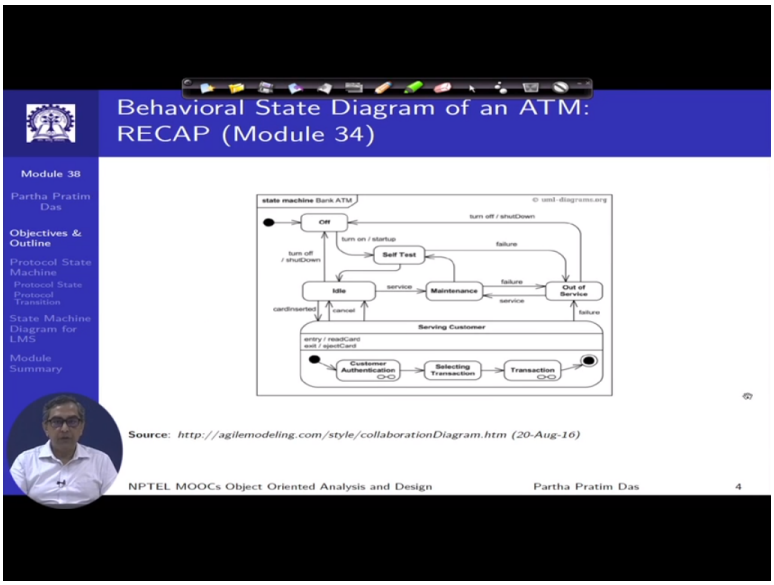
Here the state behave are often called behavioral in the sense that the states may have certain entry activity, certain exit activity and internal activities and transitions and we have seen that there are variety of behavioral states starting from the simple state, composite state and submission state and composite state states in turn could be orthogonal or non-orthogonal, that is they could be just sequential having one region or they could have multiple regions.

And we have also seen that the state machine diagrams described need variety of other kinds of nodes for its complete description and these kind of descriptions known as the pseudo state nodes which include initial final kind of node and in addition it has for, join this kind of node as well which or choice nodes which defines different kinds of control transition in the whole state machine process.
(Refer Slide Time: 02:07)



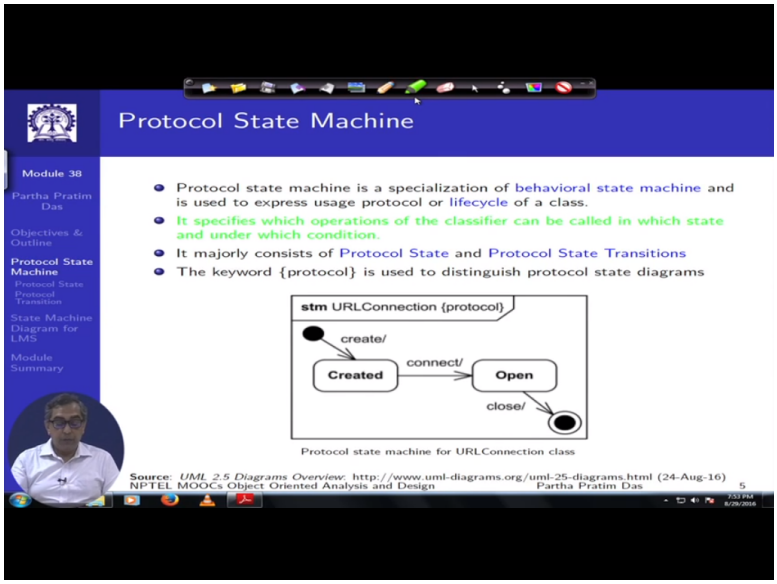
So will continue on that. This will be the last in the series of discussions on the state machine diagram. So will talk about the protocol state machine at the end we will briefly take a look into the leave management system example we have been discussing often and discover, try to discover the state machine in depth.

(Refer Slide Time: 02:30)



So this is just to recap that this is one of the behavioral state diagrams of an atm that we have done and am sure you have understood this well and if you not please go through that again.

(Refer Slide Time: 02:44)



Now first about protocol state machine. Protocol state machines are a specializations of the behavioral state machine and there they are used to express the usage protocol or lifecycle of a class. so let us understand what was a behavioral state machine. Behavioral state machine if we just take in the very

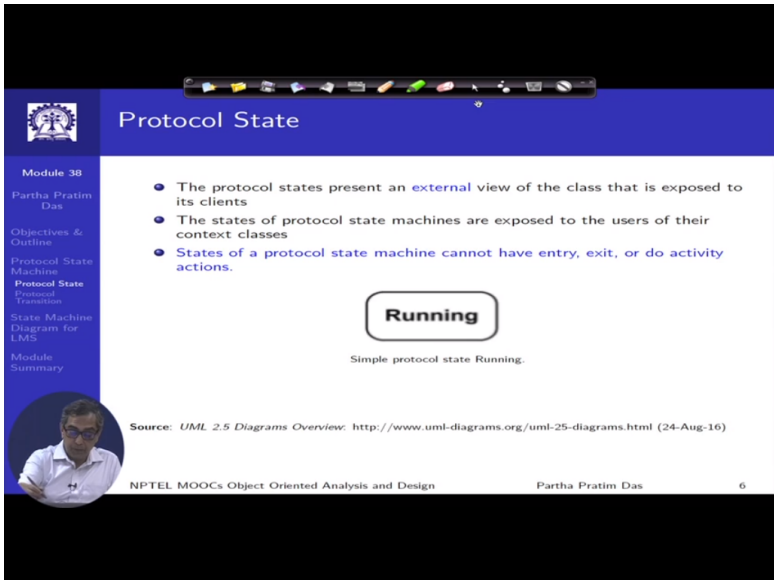
simple form I have a state s1 and I have a another state s2, these are behavioral in nature. So what we do? We have for this I have an entry, for this I also have an exit present here and I have some internal activity.

So they are behavioral states are kind of at a higher level of abstraction defining not only defining the overall conditions in which the system exits. But in terms of modeling the system at a micro level may be going through several changes because of the do activities that is happening in that state but overall we see that the system is in single state. So we have seen several examples of that. But here in we in terms of the protocol state, we are primarily interested to express the usage protocol in terms of the lifecycle of a class that is in protocol state machine.

We will not have the entry, exit or the do activity of a state. a state is more a designated of what is the condition on the lifecycle of an object of a class that the whole process is currently behaving. We will see that actually when we talk about the variety of different state machine diagrams, quite a significant part of the state machine is actually protocol state machine. It just talks about the lifecycle transitions without having further activity details amongst it.

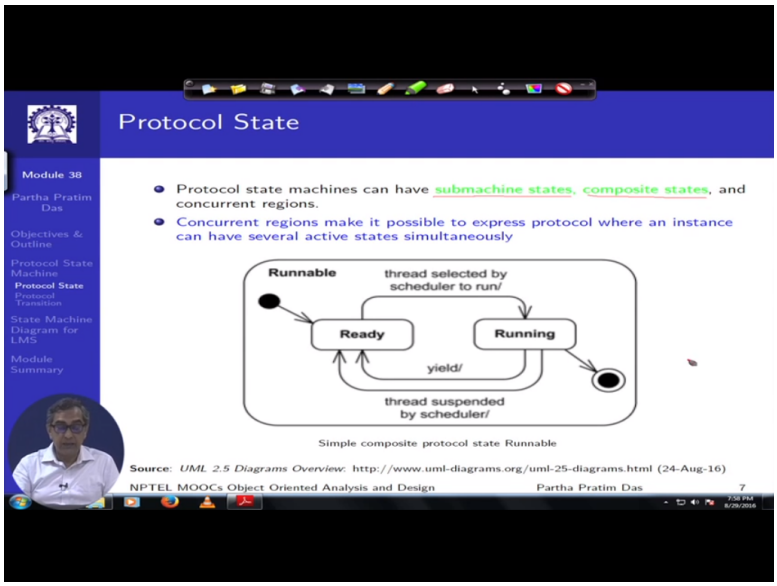
When we represent a protocol state machine we usually use engage a keyword protocol written in the curly braces which designates that this is a protocol state machine. So this is like before this involves this is a initial pseudo state, this is the final pseudo state final state and these are the protocol states. So the when you once you say this is the protocol state what you mean is these will not have entry, exit or activities within them. It just designates them this has happened. so I could actually model starting with a protocol state diagram.

And then when we get more information we could elevate some of the protocol state actually behavioral state by adding all those behaviors. But first that we need to identify is the distinct configurations into which a system should exist so far as our modeling is concerned. So we saying in terms of a url connection there is created that is you are just beginning the process and then if the connect event happens, you have been able to connect with that url and you are in open state, you are checking that url and when you close that browser, you have a close event it takes you to the final state.
(Refer Slide Time: 06:46)



So protocol states represent an external view of the class that is exposed to its clients that is whatever the client is doing with that class, you will be able to see that in terms of the protocol state. The states of a protocol state machine do not have this we have already mentioned. So this is a simple protocol state running, so protocol states will only have the names represent.

(Refer Slide Time: 07:17)



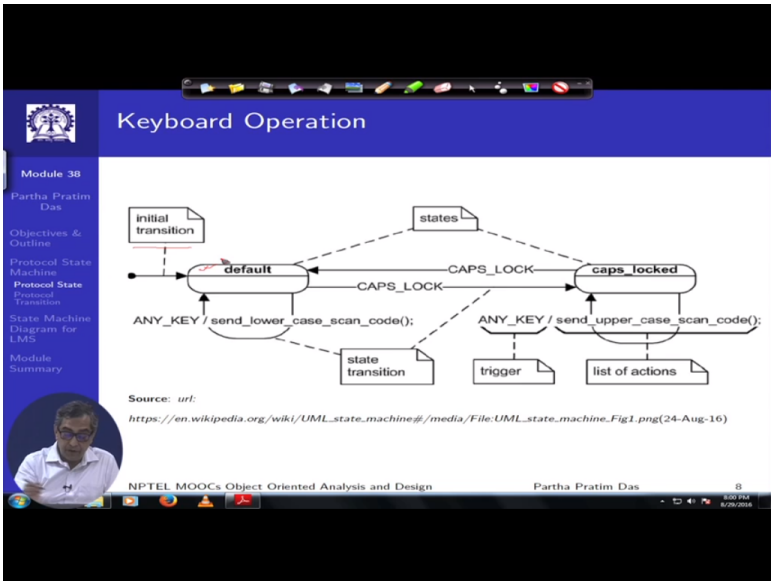
So in terms of that the protocol states also can have submission states or composite state or concurrent regions. The concurrent regions make it possible express protocols where an instance can have several active states simultaneously. So the difference with the specializations with respect to the behavioral state diagram is in the fact that protocol states are just place holder of configuration but they do not have a behavior often.

But otherwise you can have a composite state at this for example it is been shown here is this is composite state runnable and within that we have 2 protocol states sub states, ready and running. so this is part of as you can understand, this is part of the process state transition diagram that we typically draw as a finance state machine we are drawing it as a state machine uml. So you say that these are the sub states ready and running sub protocol states.

And so starting from the initial it could be in ready and if a from the ready if a particular thread is picked by the scheduler to run, you get the run event and your state changes to run. And then when the thread is suspended because you have possibly run out of your time slot then this is suspended and you get back to the state ready from the state running or the process might yield because the high priority process may come in so the process will yield and come back here.

So we can see that, likely we have seen in terms of the behavioral the state machine here too we have composite state and here too we have sub states and so on. Similarly we have submission states and we can have concurrent region where the protocols run in parallel in concurrent so that we have multiple states finishing at multiple different points in the dynamic system.

(Refer Slide Time: 09:32)

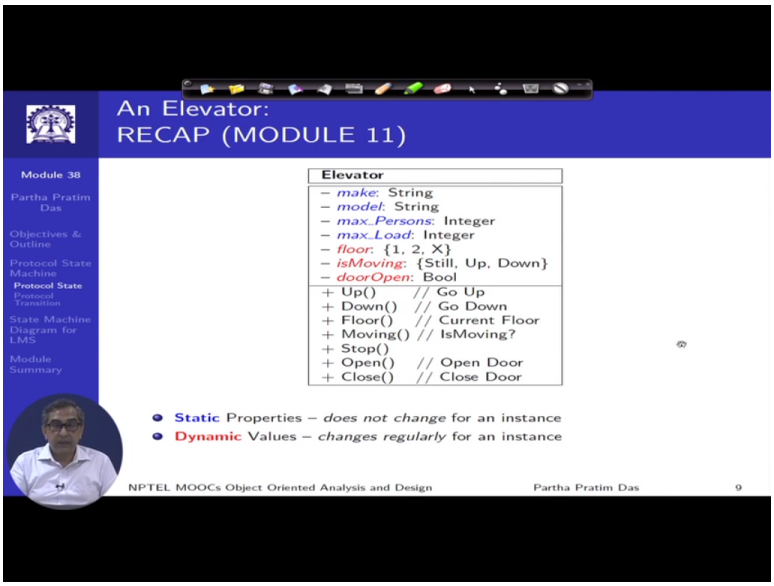


This is just a show you showing you a keyboard operation. So this is the initial transition, these are the different protocol states and so by default so these are the states that we can see and by default you are in the lower cases character mode, so if you press any key then you send the lower case scan code because you want you know that you are going to find out the lower case character from the keyboard and you continue back in the same state machine.

But the moment you press the caps lock key you have got the caps lock event then you make a transition to the caps lock state which is the sub state of the whole composite state of key board operation and varying if you press any key then you have the upper case scan mode switched on and each one of them will have the list of actions, this is the key press is the trigger that is happening for a state transition to take place.

And in this state if you have caps lock pressed again then you will again comeback to default, you will basically keep on toggling between these and these are the different transitions you can see. So this is a typical way of representing or sample showing how the protocol state machine will actually look like or operate.

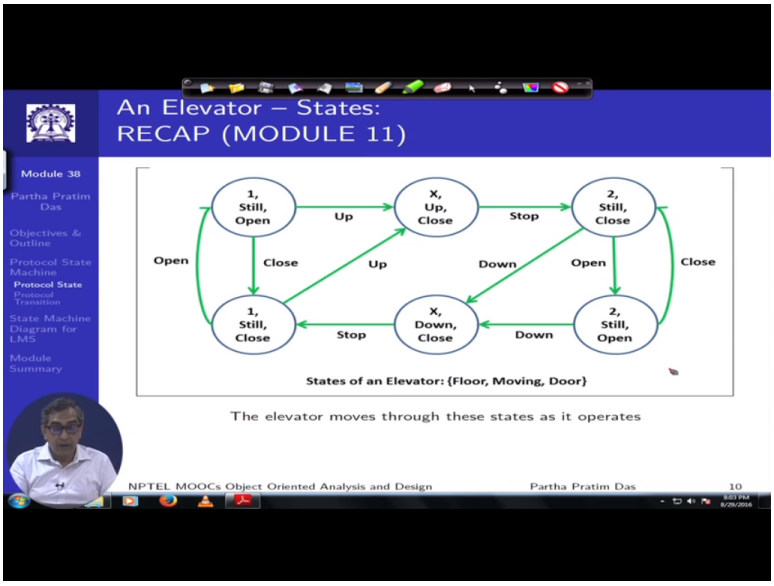
(Refer Slide Time: 11:09)



In this context I would like to also remind you about an example that we had talked about in a quite a few modules earlier talked about an elevator which that we said that elevator can work between 2 floors and we tried to model that elevator in terms of certain static properties which describe the elevator instance but more importantly we looked at the dynamic properties of it they move between one floor to the other.

So we said that the floor could have 3 possible values, so 1 and 2 the first floor and second floor and x which is indeterminate floor or the moving state and we said there is a is moving condition which defines weather elevator is still or it is moving up or moving down and the door is represented to be open or close to a Boolean.

(Refer Slide Time: 12:24)



And so based on this we could work out with a state machine diagram for this whole scenario and if you now with all your knowledge of the state machine diagram look back into this simple state transition that we had done. Then we realize that this is nothing other than a basically a protocol state machine because you have here all these different descriptions, the triplets defining what is unique about that state and these events of trying to go up, trying to come down or trying to open the door and so on, you find the different transitions.

Naturally we can also think about this so this could be the first model of an elevator in terms of the state machine diagram which is the protocol state machine but we could also make this subsequently into a behavioral diagram saying that well if it is say if it is one still open, one still open means it is on the floor 1, it is not moving, it is still and the door is open. so at this there what what will be the entry exit for this?

The entry exit if you if you see the entry for this is is open right so if somebody has pressed the open key open button on the elevator and only then you will enter this. And on exit what you will have, you will have close. You may have close and up but you will at least have close because if you take this you will have close, you will have to go. So you can see that the close is common between these both of these transitions we can say exit is close and what is the do action that this state might have is is scan keys.

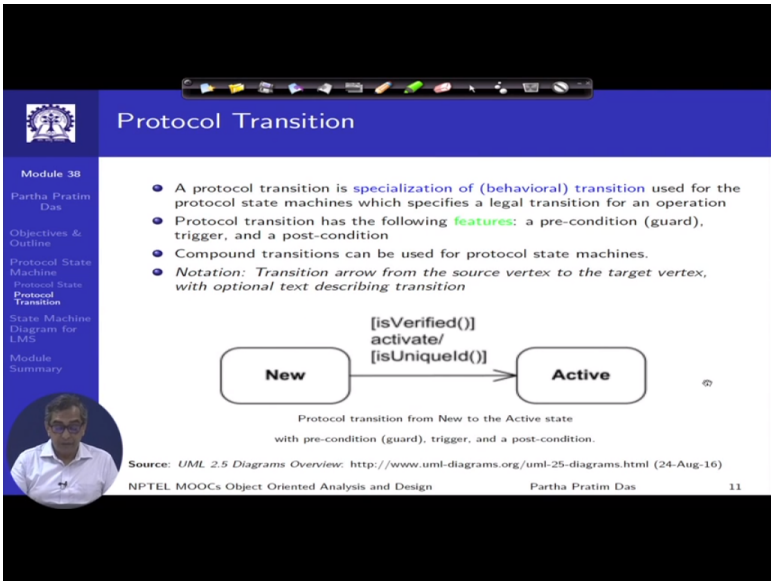
So these are different possibilities that will exist in that state and having identified that after we have

protocol state diagram which is just looking into the different 6 different configurations in which the um elevator can be we could then endeavor all these entry exit and activities and try to bring out the real behavioral state machine out of this. For example another if you if you take so if you consider this state, this state is x that means the floor is indeterminate.

Because the moving is up that is it is moving up and the door is closed. So which means that the entry is clear, entry is up, exit is clear, what will happen in exit, it will stop and what actions have been taken, it is moving right so this is going up, so this activity is continues to happen in this state and whatever else need to happen because of that because the motor has to run, the pulley has to turn in certain direction and so on.

So all these if you try to describe along with the state description here, then you will elevate this protocol state machine into a behavioral state machine. So that is why we started by saying that protocol state machines are just specializations of the behavioral state machine which give you a compact information in terms of the lifecycle of the objects of the class. So you can you can really see what these these different lifecycles are happening and they will keep on continue to move around in between this protocol state states and will give you information in terms of how the system grows.

(Refer Slide Time: 16:21)

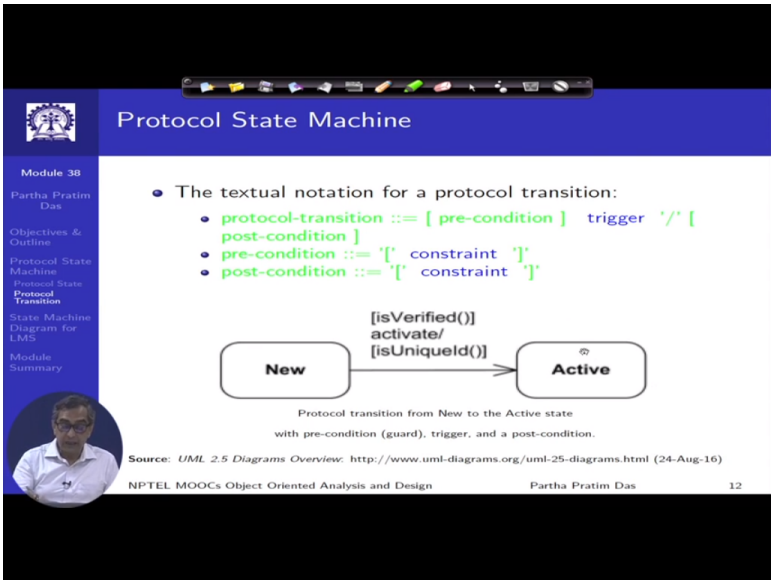


Protocol states do have naturally have transitions and often the transitions are guarded. So a protocol transition is again a specialization of a behavioral transition and used for the protocol state machine and has a following feature which is a pre condition or trigger and a post condition. Pre condition, trigger and the post condition. So precondition is what must satisfy before this transition can take place, trigger

is what makes the transition happen and post condition is what must be satisfied once the transition is taken place.

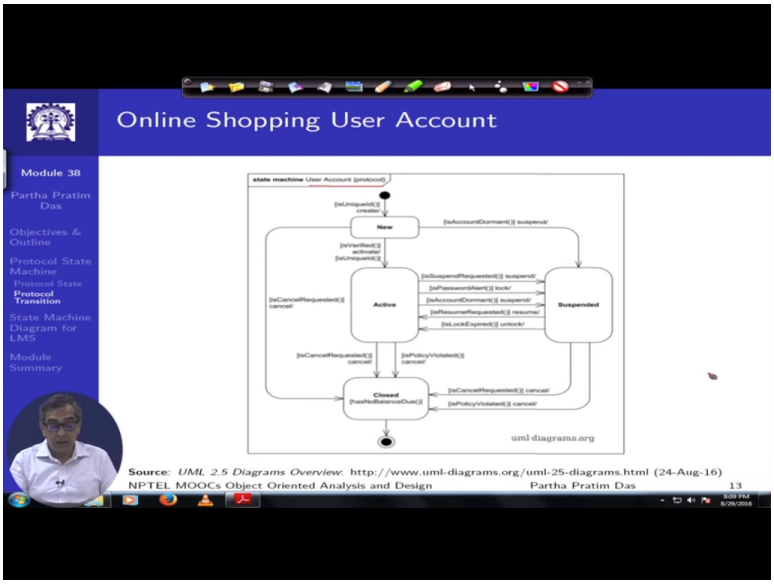
So this concept we have been saying repeatedly in different parts of uml we have seen this similar concepts in activity diagram, you have seen this concept in the behavioral transition as well. So here if we see if it is, if we are moving from a new to active state then the precondition is it verified that this is a this is a valid id. If it is then the trigger is activating that id and certainly the post condition is what will result from this is unique id.

(Refer Slide Time: 17:45)



So this are the typical ways you will write protocol transitions and so this is the syntax for the protocol transition that you can get satisfied yourself certainly in terms of the protocol transition, the pre-condition and the post conditions are usually constraints. That is basically they are Boolean conditions which are true or false telling you weather the pre-condition is satisfied or the pre-condition is not satisfied, weather the post condition is satisfied or it fails.

(Refer Slide Time: 18:16)



So you could look at a protocol state machine diagram for online shopping and here I have picked up an example where you can see really the different kinds of protocol transitions happening here. So let us look at one two of this, this is unique you start here. So this is the initial pseudo state and then you have the new sub state or sub protocol sub state so the pre condition is unique id because and then you create that naturally only if it is a unique id it can be created.

And then from this state you make a transition to the suspend state, what is the pre-condition, is the count dormant. So if the count has been dormant has not been operated for a certain period of time, then you should figure suspend and put it to that makes the tran the system go into the suspend state. Similarly in the in the new id if it is verified we have just seen then the trigger and action activate and get the unique id and you reach the active state of the system.

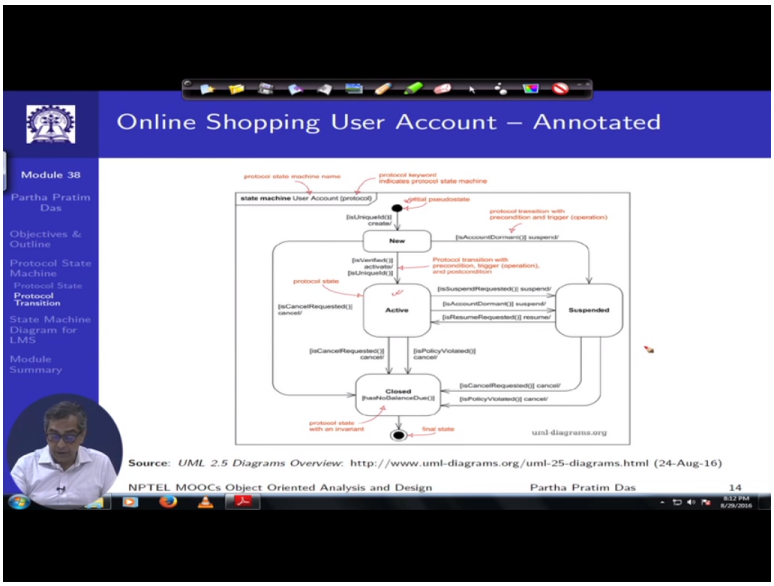
In the active state of the system again you could explicitly ask to suspend your term. to do that we reach the system. If the password alert, for example if you have hidden the wrong password for a number of times then to lock and you get in turn suspended. So all these you are expressing all those semantic you can express in terms of the protocol transitions using the pre condition constrain and the post condition constraint as applicable.

For example here you can see that you have a transition from suspend to active which is based on the pre-condition that is lockexpired. For example when the password is locked, typically it will not be locked eternally it will be locked for couple of hours, for a day or so and beyond the expiry of that this condition gets satisfied and need to get trigger to unlock so you come back to the active state. So in this

way you could follow this whole diagram very carefully to see different situations.

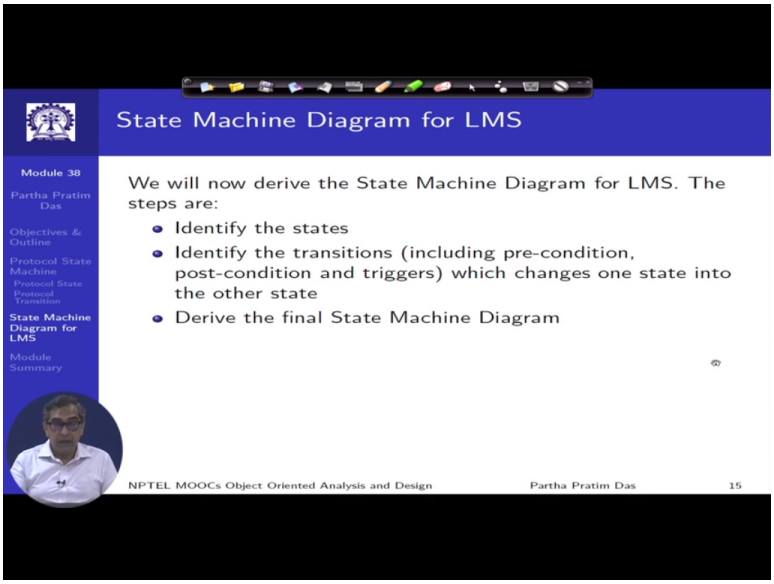
And how they are modeled in terms of the protocol state transition using the pre-condition, post condition and trigger models and learn more about how these state diagram modeling happens. These are same example where there is a little bit of annotation done so that you have ease in preparing for understanding of what these different icons are.

(Refer Slide Time: 21:09)



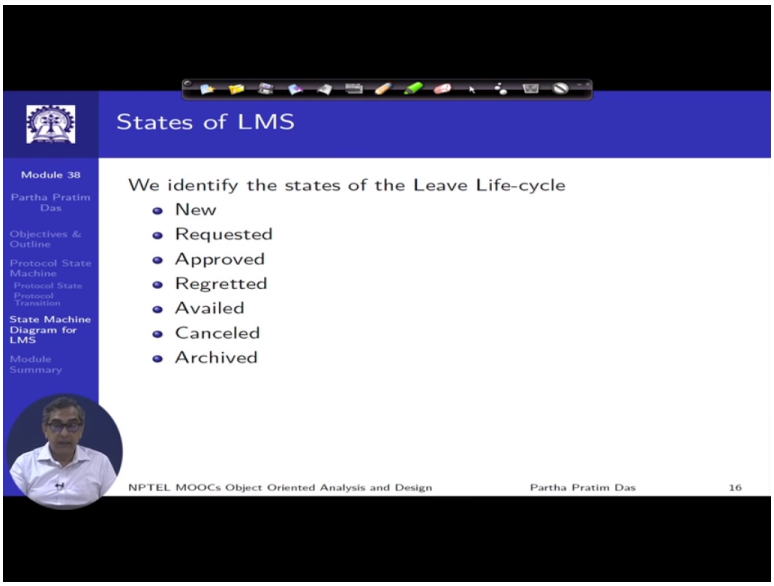
This is as you know the protocol state is a this we have seen several times. this we have just explained the protocol transition, this is another protocol state and this is a protocol state with the difference. It is a protocol state with an invariant which says the count is closed, has no balance due. So if that is the case, then only in that is will get closed. So this is the invariant which is important for this state.

(Refer Slide Time: 21:43)



So we will had to, before we end we will take a quick look into the state machine that may be important in terms of the leave management system. We certainly need to identify the states, the transitions and the state machine diagram which we will start with by having a protocol state machine diagram.

(Refer Slide Time: 22:02)



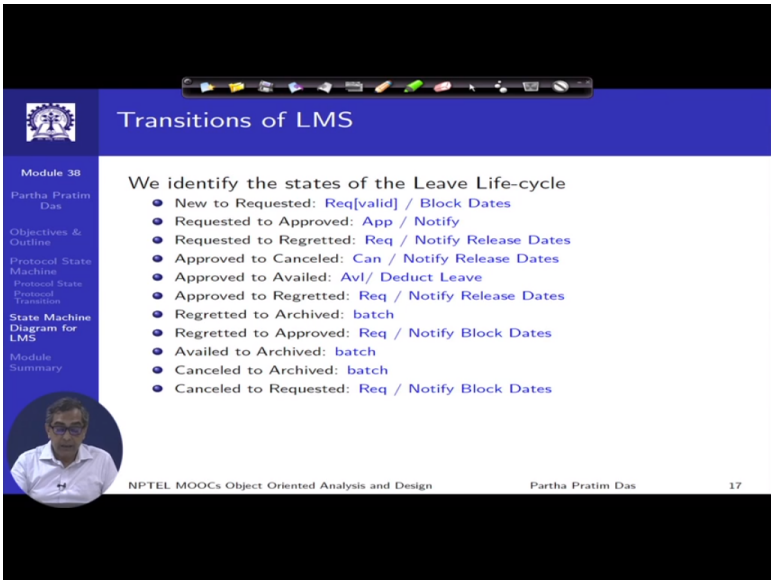
So what we identify is we have seen that leave is a first created through a request and then it goes through an approval process and it may be approved, it may be regretted. If it is approved then it may be availed or it may be cancelled, if it is availed then it will be redacted and so on. So we can model this whole information in terms of a lifecycle of a leave. You can consider the leave as a lifecycle which has these different kind of states.

So we are just trying to in terms of building a protocol state machine we are just first trying to identify

what the different states could be. so when the leave is just first created, you recall that while any request is put in, you need to leave has to validate itself and make sure that the leave being requested is a valid leave. So that happens if it is just a new leave and after the validation when the request has been I mean when the request has been registered in the system not approved yet.

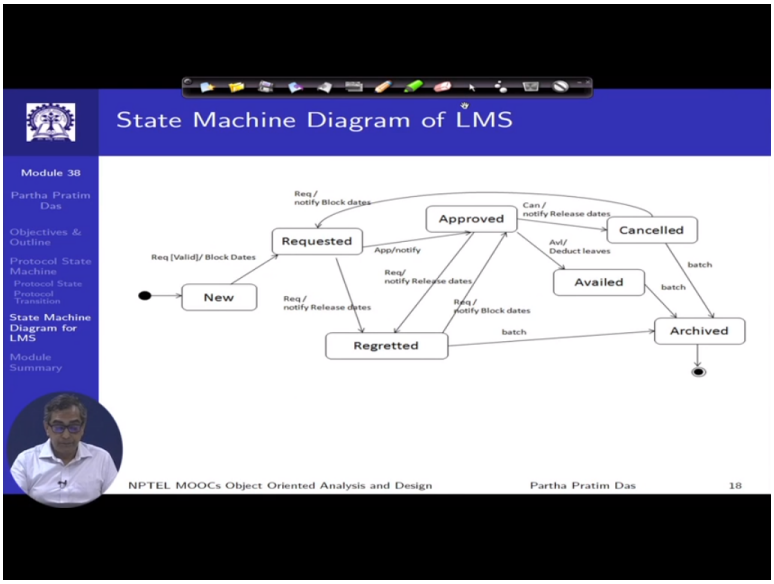
When it has been registered in the system as a valid leave, then it could be in the requested state and based on the request the lead or the manager may or may not approve and therefore it go to the approved state or regretted state and so on.

(Refer Slide Time: 23:39)



So if we just follow that then we will be able to identify that different transitions for the leave lifecycle like in terms of new to requested, requested to approved, approved to regretted and so on. So basically once you have identified the stage you look at, what are the transitions that are possible and what are the triggers and the pre-conditions for that if that can be defined and you try to through that.

(Refer Slide Time: 24:09)



So this is where I present the leave system in terms of the state machine, this is the initial state and certainly this is the state where the leave has been created. So new and then based on this you have a request so what is what is this transition, in this transition will be possible if the new leave is a valid leave. right, if the leave is not valid then this is not right. Then we write in that notation then if this has to be a valid leave and a trigger is a request.

What is a post condition, a post condition is blocked, blocked dates. Why? Because if somebody has requested for a leave say from 1st June to 3rd June 2016 then once the request is registered that is what we are designating by moving the leave to the protocol state requested, then the those dates must be blocked so that no further leave request or other request can be made on those dates. So we can clearly see that as this transition happen from new to requested, this is the pre-condition that you need. Trigger is the request and this is the post-condition that must happen.

So in a similar manner you could continue on that if a leave is requested then it may be approved, it could decide further this is just I have put just a cautionary you know transition information you could put more transition information, it could have multiple more states here depending on the different situations of the approval, some approvals are conditional depending on the validity or availability of the supporting documents, some are direct approvals and so on.

So you could decide, you have more protocols states describing that and then have more pre-conditions and post-conditions defining that or otherwise the requested leave could be regretted so any of these will have a post condition of notifications whether its approved or regretted, then the notification that

will be done and the leave will be in one of these states. An approved leave may be cancelled would go here, it may be revoked actually you could have this is regretted it should not actually happen here.

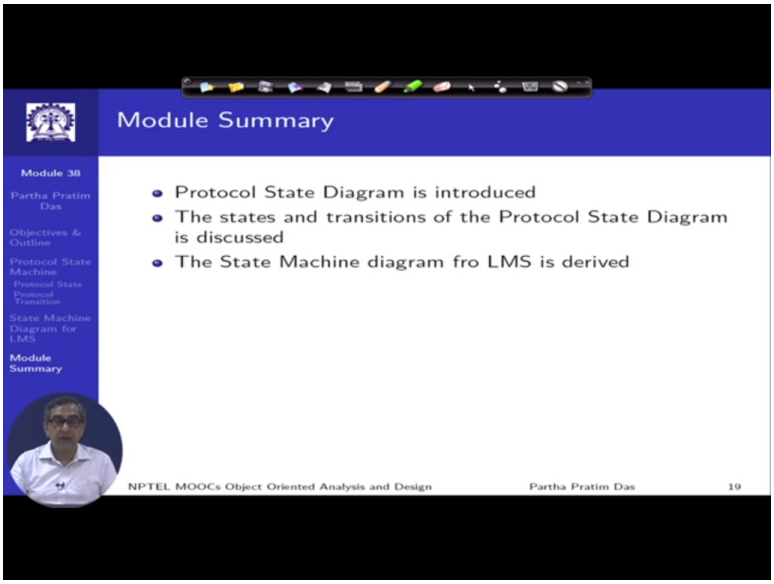
The better way to model could be you can have revoked state protocol state. so I can the manager could revoke an approved leave and a revoked leave cannot be availed in further. An approved leave can be availed and then any of these if it is an achieved leave, it is a request regretted leave, a revoked leave or an availed leave could actually go into some kind of a you know we can say a am in search of a word, a dead leave should have no further changes can happen to that leave record, which is then archived in the system.

So this is just an outline showing you what are the different transitions that can happen. if you analyze further will this is and this just gives you the protocol state diagram. So in each one of them for example the requested, you could now take make this protocol state into a behavioral state by defining the exact entry, exit and with the 2 activities here because if in the requested state, before the transition happens you have activity where the manager.

Or the reporting manager of the requesting employee has to look at this leave consider all factors including the documentation and other conditions and then take a decision on that. So those could be the activities in each one of these states that we identified and start encoding put into the state machine diagram more and more you do that, this will slowly become more complete behavioral state machine diagram of the leave management system.

So this is just to show you that even such a simple system as for leave management has a lot of scope there this does not immediately look like a problem, I mean is not an electronic problem or a or a physical problem like the temperature controller or the calculator where the states look to be more oblivious but here you can see that the dynamics of the system again depend on a set of states which is very typical for any of the complex systems that you would like to model uml.

The state machine for that purpose or state charts for that purpose turn out to be really useful tools in that modeling state.
(Refer Slide Time: 29:27)



To summarize continuing on the discussion on the state machine diagram you have in this module discussed about protocol state machine diagrams, we have talked about sta protocol states and transitions along with that pre-condition and post-condition and you have taken a look into how LMS the state machine diagram for LMS can be constructed.