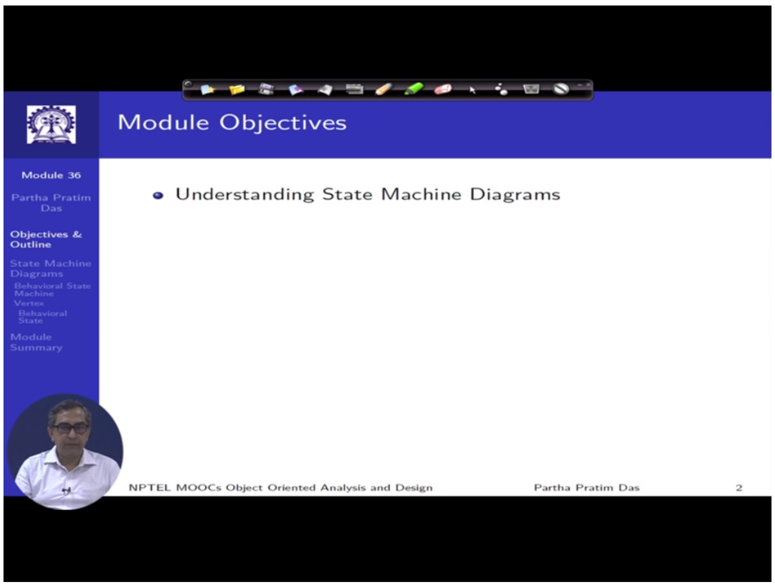
Object-Oriented Analysis and Design
Prof. Partha Pratim Das
Department of Computer Science and Engineering
Indian Institute of Technology-Kharagpur

Lecture – 48
State Machine Diagrams: Part I

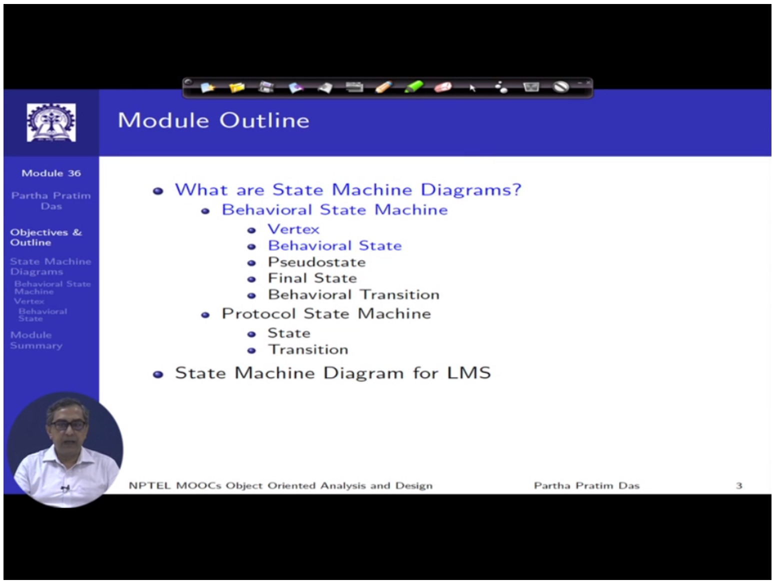
Welcome to module 36 of object-oriented analysis and design. We have been talking about different uml diagrams. In the current module and continuing in the next 2, we will talk specifically about state machine diagrams.

(Refer Slide Time: 00:37)



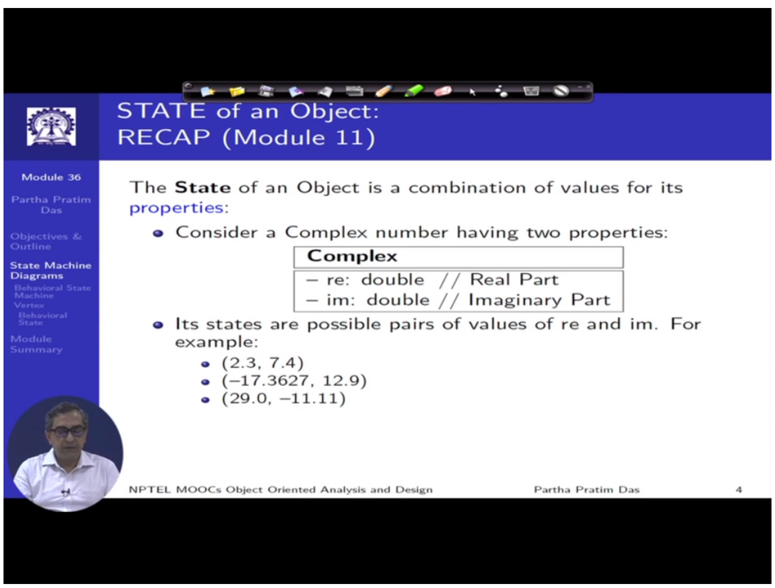
So the objective in all these 3 modules would be understand the state machine diagrams and accordingly this is the oultline that we will follow.

(Refer Slide Time: 00:51)



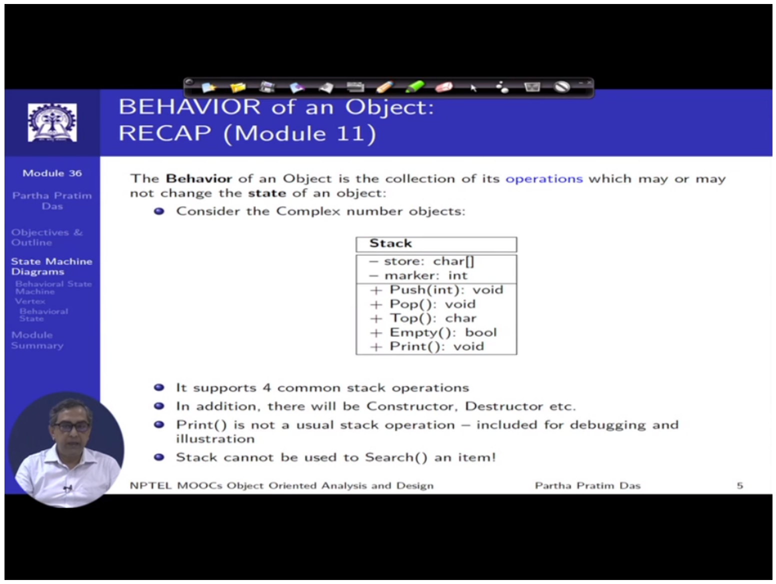
And have as we have been following the blue part on this outline is actually what we will do in this current module. Remaining parts will be done in the subsequent modules.

(Refer Slide Time: 01:07)



So to start with we will make a quick recap in terms of some of the basic concepts which will be referred more frequently in this module. First naturally the concept of the state of an object. So we have talked in terms of an object being defined in terms of the states, its operations, its identity, particularly if you talk about state depending on the different properties that it has then different combination of values that the properties can take define the state object. And we have seen that it could be static or it could be dynamic.

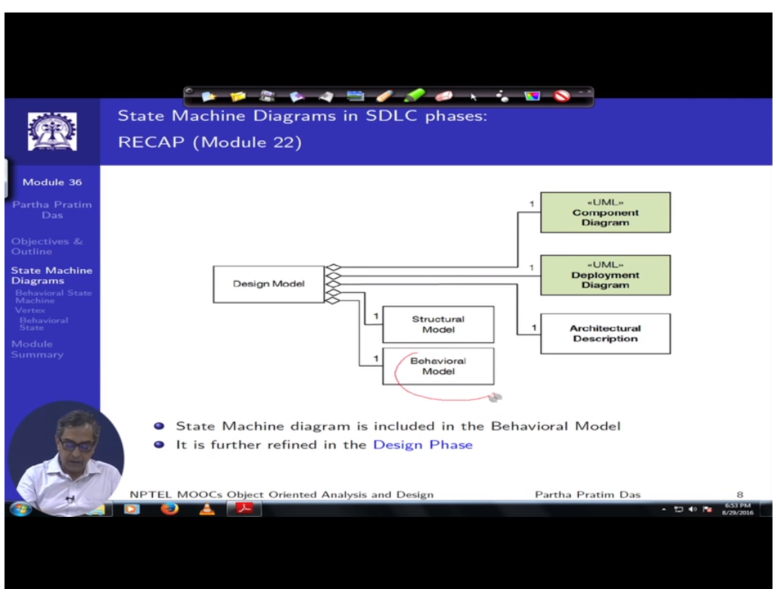
(Refer Slide Time: 01:52)



We have also seen that based on this there is a behavior which is based on the combination of operations which certainly are defined based on the state of an object. We continue to refer to the client-server model am sure by now you will understand the client server model. In terms of the sdlc phases, software development lifecycle phases, a state machine diagram is first constructed in analysis phase.

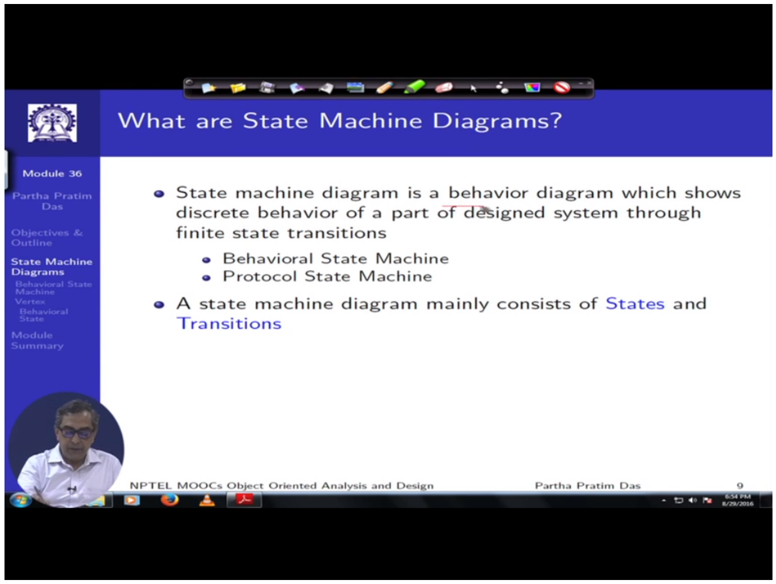
It is also called the state chart diagram; actually the earlier name was state chart diagram, in the recent versions of uml this is more frequently called as a state machine diagram. So in the analysis phase as we analyze different sequence and collaboration information and we talk about activity, we also extract the state based information from the state machine diagram.

(Refer Slide Time: 03:03)



And as is true for most of the other models we refine on this state machine diagram behavioral model and had get more accurate state machine diagram for the problem at state.

(Refer Slide Time: 03:20)

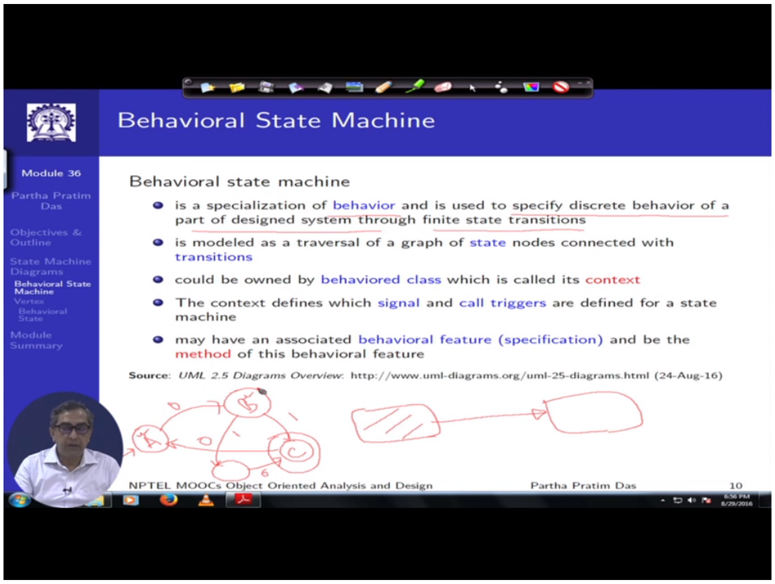


So this is the basic background and based on this we would like to talk about the state machine diagrams. The state machine diagram is a behavioral diagram; it shows the discrete behavior of part of the design system through finite state transitions. So quite a lot is being said here, one is let us look at these in parts. one is this is a behavior diagram, we understand some kind of a behavior and it shows a discrete behavior, discrete behavior in the sense a behavior could be continuous as well.

But here you have discrete behavior that is it is defined in terms of certain set of discrete values and therefore the combination of this is values define a finite combination which we think as a finite set of states and their transitions. And there are primarily 2 kinds of state machines that the state machine diagram would be involved with. 1 that is called the behavioral state machine and the other is a protocol state machine. Naturally a state machine will consist primarily of states and transitions.

So in that sense they are pretty much like though they are substantially endowed further but they are pretty much like the finite state machines or the regular automata that you have studied in other courses. But here they will take much more complex shape.

(Refer Slide Time: 04:52)

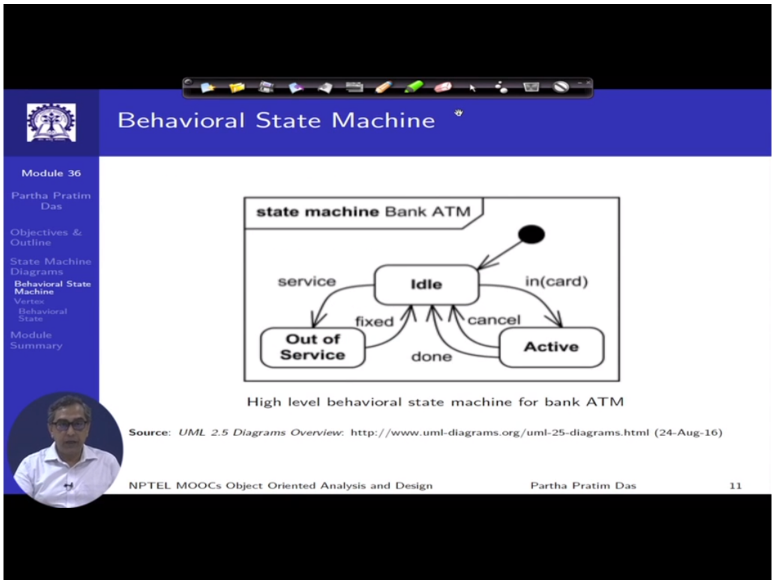


So first let us talk about the behavioral state machine in this module. This is a specialization of the behavior and is used to specify the discrete behavior of a part of the design system through finite state transitions. So just think about what you are saying, there are there will be states, say these are states and there are transitions between these states. And every state is a has a certain behavior. So unlike say the traditional fsm if you think about you would have done traditional fsms like this with 01 input you have state a, state b, you say on 0 go to this state on 1 come to this state, on 0 go to this state and so on.

You you must have seen such kind of state machines like this and some states are defined as final states, some as an initial state. Here in so states in in the traditional fsm sense are basically place called that markers which remember the kind of discrete combination of information that the particular represents. But here we will see that in place of a behavioral state diagram, the states are more empowered. They have a behavioral feature or specification, in that a state will may be have certain conditions or event for entry, and every state may have some entry.

Every state may have certain conditions or actions on exit and typically they will have a behavior that happens in this state. So it is this is not a state at a very micro level of the system description. They could be at a much higher level of abstraction where a whole lot of things might happen within a behavioral state based on a certain entry and certain exit and that behavior is what will be captured in the action for this state. So the state in the in this context will often be owned by the behavioral class which is called the context of the state and which defines the signal and call triggers that will make an entry into the behavioral state of a behavioral state machine.

(Refer Slide Time: 07:30)



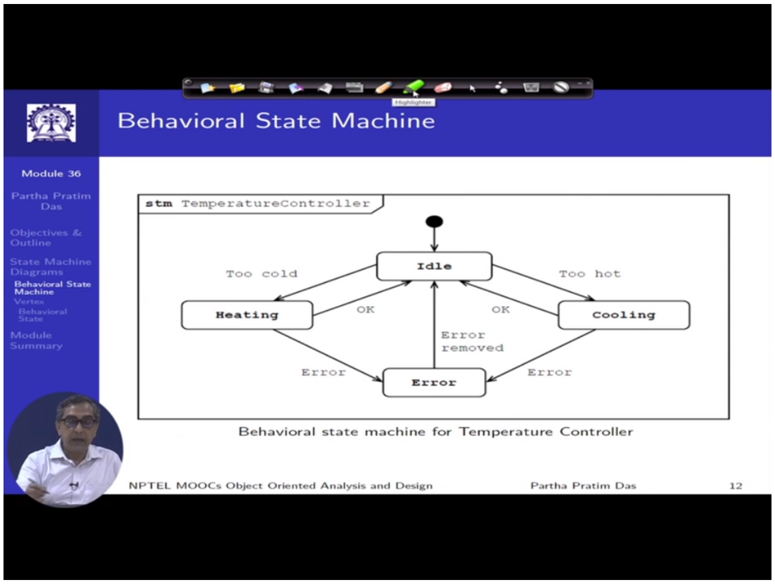
So let us slowly start taking a look into some of the possibilities. So we start with the simple one. So this is a state machine of a bank atm possibly. So this is where you start and when you start, the machine is in an idle state. Now this idle state is a behavioral state in the sense that while it is idle it is not a passive state it is in, it is still active to receive inputs, appropriate inputs if we keep on checking if somebody is pressing some buttons, somebody is inserting a card or anything like that.

So at a macroscopic level, this is an idle state, but internally it may be doing quite a few things. but then if you have a specific event like in card which means a card is entered, then it gets into another state which is active state where this is a state where it is supposed to do a whole lot of things basically having an active will actually interact with the user and possibly allow the user to withdraw money or check balance or things like that and when all that is done, then it will come back to the idle state.

Or it could be that in the half way either the card is not accepted or the pin was wrong or the user decides to abort the transaction that the user was doing, then it could be cancel and it will again come back to the idle state. An alternate path could be that the machine is in idle state and it needs be, services needs to be maintained certain preventive maintenance would be done, so it is put into the service mode. Even it is put into the service mode; it gets into a state which is outer service state.

And only when the maintenance or servicing is over, then it will again have a transition takes which will bring it back to the idle state. So we could in terms of very simple three states we can attain abstract high level, we can describe a atm application and atm process and this is what is the purpose of a behavioral state machine. There could certainly be lot more details than this, we will get more details but let us work through couple of other examples.

(Refer Slide Time: 10:04)



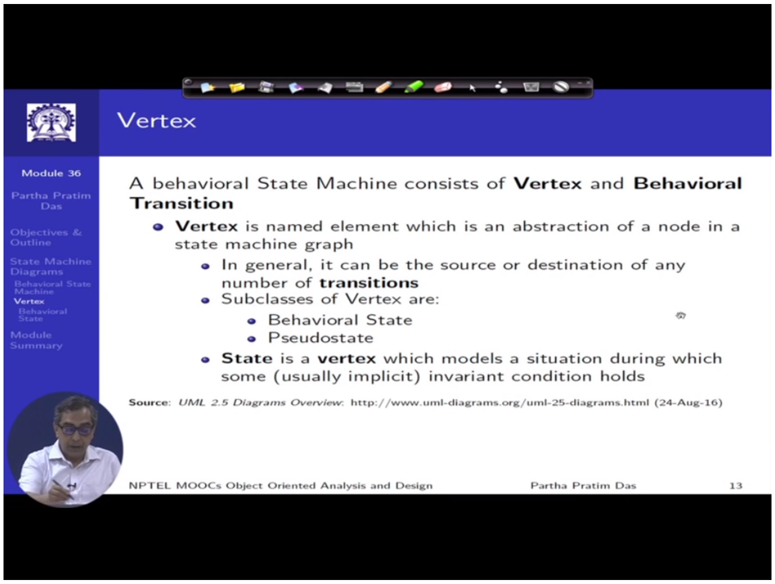
This is another example of a temperature controller which again starts am sorry this which again starts in the idle state and again what I am trying to repeatedly point out, this is not like just the finite state machine where if it is idle state it will it will basically continue to be in the state unless some specific input comes in here, in the idle state itself it is doing a lot of actions, it is possibly trying to sense this is the temperature controller it is regularly collecting the current temperature value through the temperature sensors and based on the value of the temperature if it is too cold.

It changes state and puts on the heater. So when it state is heating again lot of things are happening in this heating state, the heater is on and again the temperature is being monitored and so when it is co become adequately hot or less cold then it says ok, heating is done and it comes back to the idle state. It has achieved the temperature that has to be managed. similarly it can happen on the other side there is the temperature goes higher than what is specified then it will get into this cooling state, put the cooler on and so on.

And on completion, on achieving the target temperature it will come back to the idle state again or it is possible that from the either the heating state or the cooling state, there might be some error, the heater might malfunction, the cooler might malfunction and it might go into the error state and it will come back to the idle again when the error is removed. So each one of these again at a high level describes the overall behavior of the system.

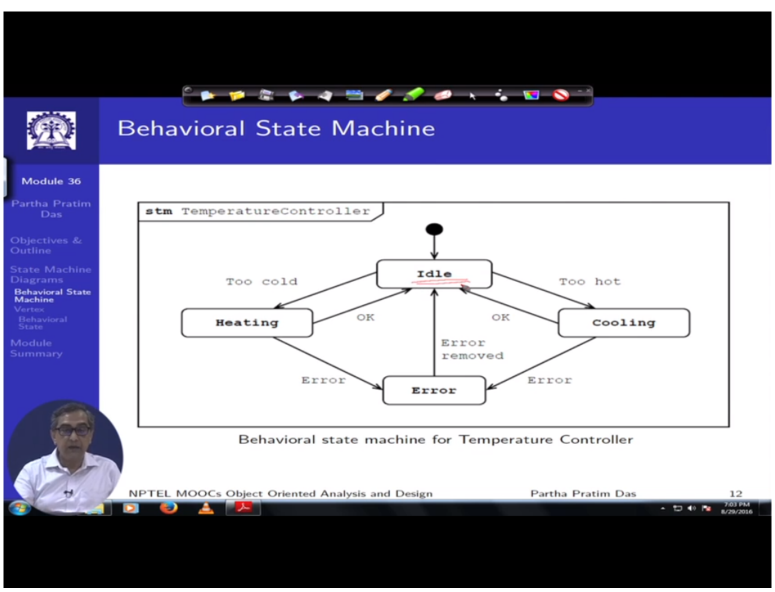
So when we capture the system requirements and start analyzing, we try to see the try to find out situations where we try to identify discrete states, behavior states of the system which can in crisp way describe the overall discrete behavior of the system that we are trying to describe.

(Refer Slide Time: 12:04)



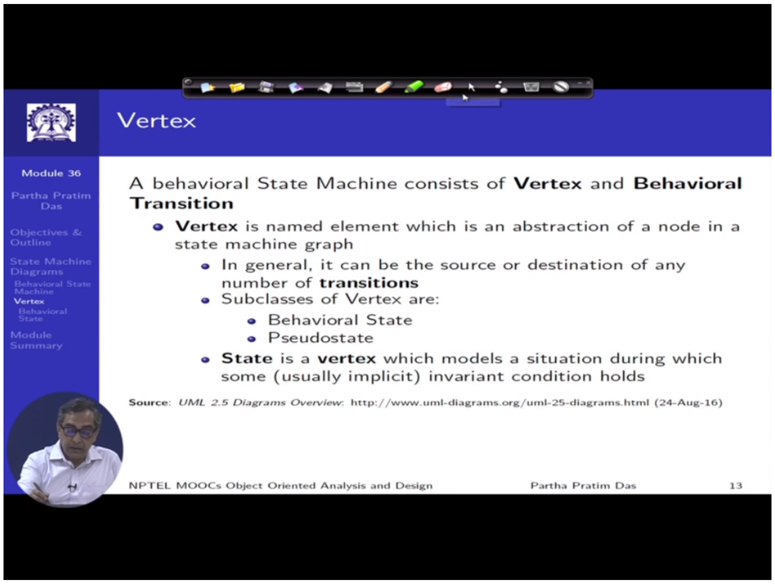
So the behavioral state machine in structural terms consists of vertices and behavioral transition. A vertex is a named element and which is abstraction node in the state machine graph, we will see there are other kinds of states as well. And there are primarily 2 sub classes of vertices; one is the behavioral state vertex and the pseudostate vertex. So state is a vertex that models the situation during which usually some invariant conditions hold.

(Refer Slide Time: 12:51)



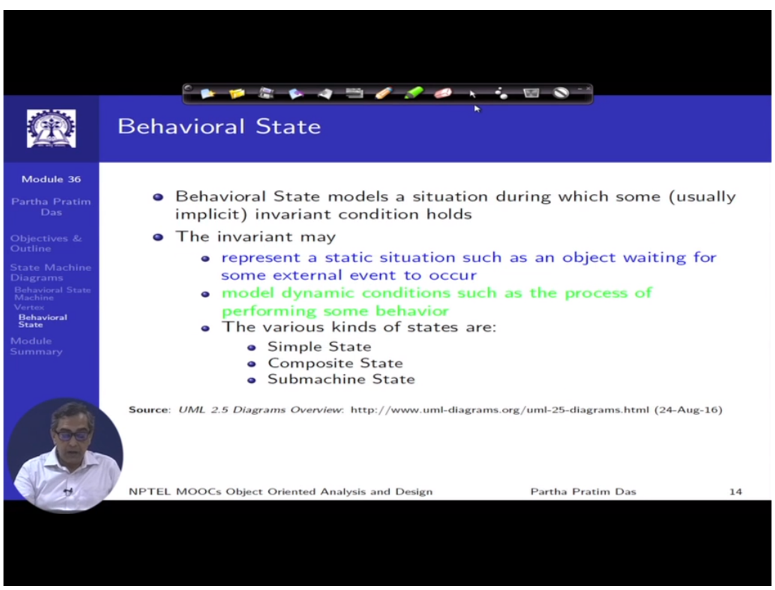
So you if you just refer back to let me just take you back to the sorry let me just take you back to here. So this is a state because here certain invariant condition hold, the invariant condition is temperature is within the desired limits. This is where some other invariant conditions hold, this temperature continues to be remained lower than where it should be and the heater are actually operating and so on.

(Refer Slide Time: 13:17)



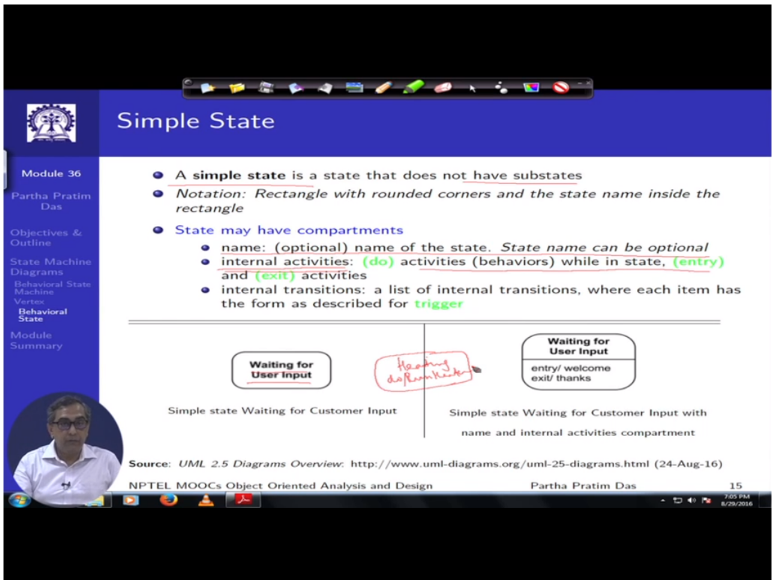
So that is how the state will be a vertex and so what you saw there are the different vertices of a behavioral state machine diagram.

(Refer Slide Time: 13:25)



So the behavioral state ss there are first of the 2 kinds of vertices that you have, the behavioral states has invariants as we say they represent a static situation such an object is waiting for some external event to occur or certain or particular condition is being maintained and it modeled dynamic conditions as that the process of performing some behavior like it is in terms of the temperature controller it is the Process of heating which is regularly performing the heating operation to try to achieve the target temperature.

The behavioral state in turn will be of again 3 kinds, they are called simple states, composite states and submachine state. Let us look into the definition and then we will explain what they mean.
(Refer Slide Time: 14:15)

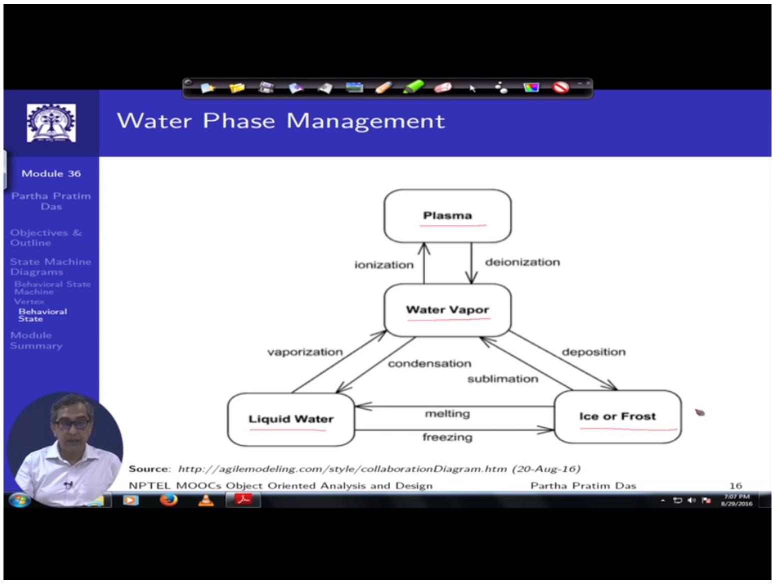


A simple state is one that does not have sub state. So we need to look at what are sub states. Will come to that when we go to the composite state. So a simple state is just a state, so it has a name that is a minimum so which describe what that simple state is doing. But in addition, so the name is is expected but it may be optional, it may not be there. but we are the 2 are the things is what is a internal activities that might happen in this state, that is the behavior has to where an object is in this state what does it do.

So if the object is in a state whose name is heating then they do action could be run heater. So what is which is activity that happens during while the system is in that state, so that is the do part or the activity part of a simple behavioral state. besides we will have an entry may optionally may have entry and exit activities. For example you are waiting for input user input is one simple state and your entry is welcome. So the activity when you enter the state is you put on a welcome message and the activity when you exit the state is you put on a thanks message and so on.

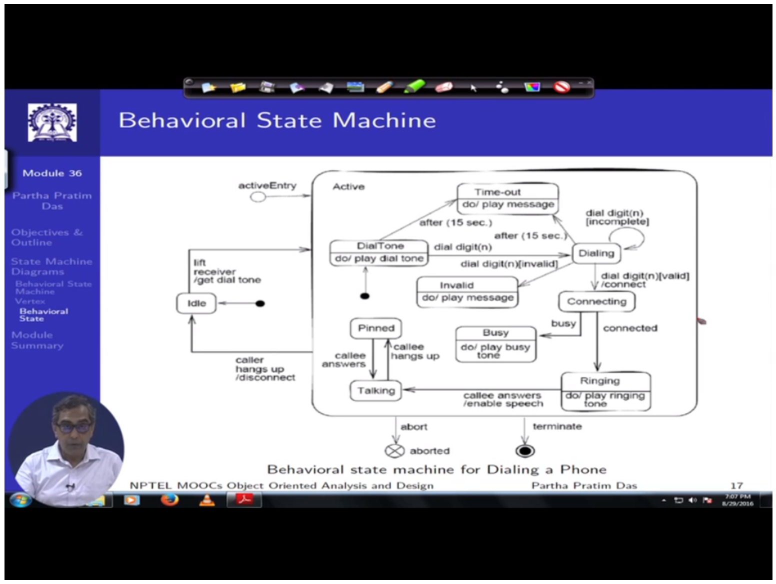
So what is important to understand and that is true for the simple behavioral state and actually in turn is true for any behavior state is unlike the state of an fsm you just remember your internal condition as to where you are in addition to that it has an entry activity, it has an exit activity and it has a performance activity, internal performance, internal transitions. So this is the behavioral state will always carry with a certain behavior which is defined in terms of these internal activity and internal transitions that may happen in interns of within that state machine.

(Refer Slide Time: 16:46)



Now so this is a just another example, so here you have multiple states, these are the 4 4 states, these are simple states and these are the different transitions. So if you have water vapor if you condense, it will become liquid water, if you freeze it will become ice, if you sublimate it becomes water vapor directly and so on. You can just follow and form a simple knowledge of physics you can understand that this kind of system can also be described in terms of a state machine diagram.

(Refer Slide Time: 17:18)



Let us look into something which is little bit more closely to the definition of behavioral state machine. This is basically for dialing a phone, so we can see these are the different states that a phone could be in. So in every state if you if you notice there is a name of the state, these are the name of the state. So this is the dial tone, this is the state invalid, this is the busy, this is the dialing and so on. So these are the states that a phone is expected to be in.

So if it is in a particular state then it has a do or action activity of that state. So if the phone is in dial tone state, then you are the activity is it will play the dial tone. So that is what we expect right, if we pick up the phone and that is in the dial tone state if we do that and in which you can, in this state you can dial the digits and once you start dialing the digits you get into different state which is the dialing state where it could dial further digits and keep on remaining in that state.

It could be that you dial a dial a wrong digit or you dial a number which is not valid, then will take you to a different state which is invalid state where the action is to play the message saying that you have dialed a wrong number or something. Otherwise if you have dialed a valid number then your next transition happens to connect and that takes you to connecting state for the phone where the number is being connected. And there are 2 possibilities from this connecting stage.

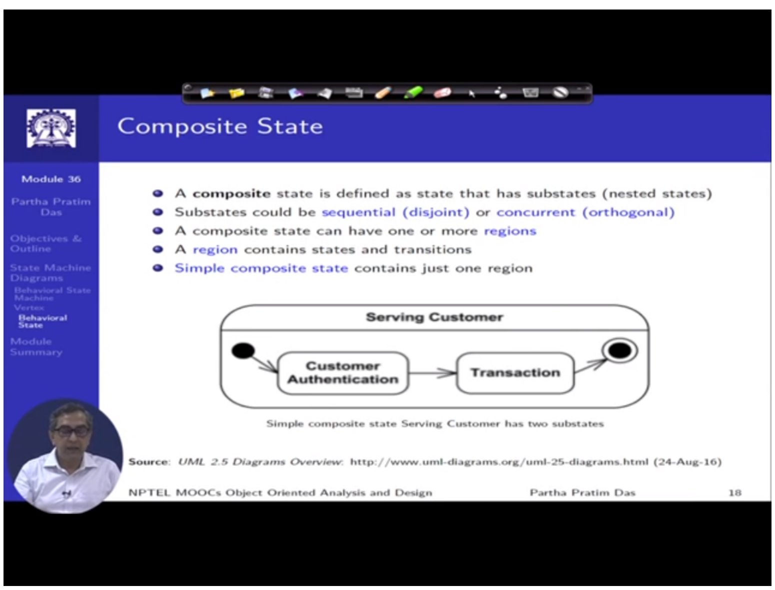
It might get connected where you might go to the ringing state where your action is play ringing tone or it could be the number could be busy then your action is get then you get into a transition into a busy state and your action is to play the busy tone. Again in the ringing state if you are in then if the callee answers then you get into the talking state and then you continue in that talking state till callee hangs up or get when the callee answers.

And this is how this whole state machine will define the different distinct you can you can understand that if you think of the phone is of a digital system then there will be several other states involved which will define how to play the dial tone, how to actually do the dialing, every key press could involve kinds of different states and so on. But we are always interested to model the system at an appropriate level and at this level if you are trying to model the dialing of a phone or making a call.

Then we are most interested to identify only those distinct states that we will the system will take and the actions that will happen and what are the transitions that will between these states so that you get a finite behavior on this. Besides that you have some you may have an idle state when you are not even trying to dial so in a state somewhere you actually entering the dialing dial tone state when you lift up the receiver and a state where you basically get back into when you hang up.

So this is a a em simple diagram showing how you can use the simple behavioral states to define a more complete state machine diagram state machine behavior of a system.

(Refer Slide Time: 21:20)

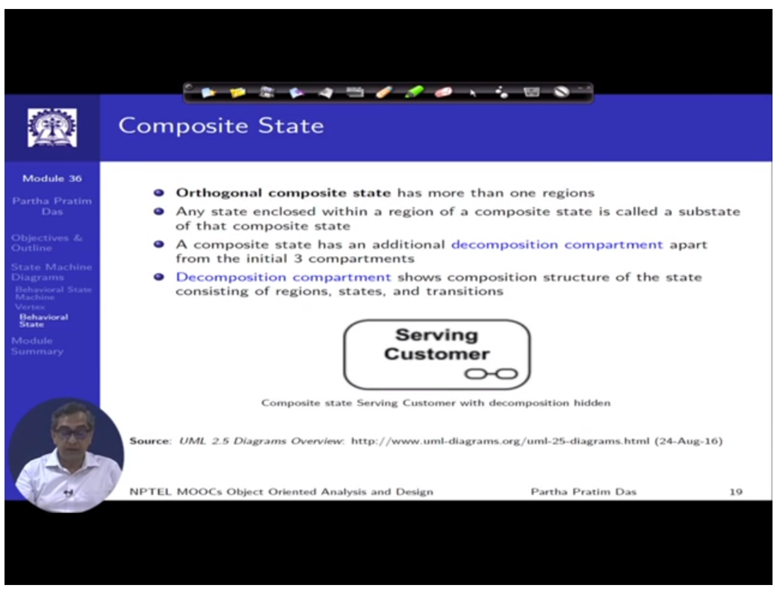


Now often we will find that the if we just want to describe everything in terms of simple state machine then we might find that the whole thing turns out to be quite complex or quite laborer and we might just want to combine couple of different states and their transitions together into a bigger state which is called the composite state and when you are in a composite state, then all the internal states that comprise the composite state or that build up the composite state is known as the sub state or the nested states.

So here we are showing that this is the composite state which again has a number, has a name the servicing serving customer and serving customer has a customer authentication state and a transaction state and within that again lot of things could be happening but this as a whole is now a state where you say that this is where I enter and this is where I finally exit and I have sub states of customer authentication and customer and doing the transaction.

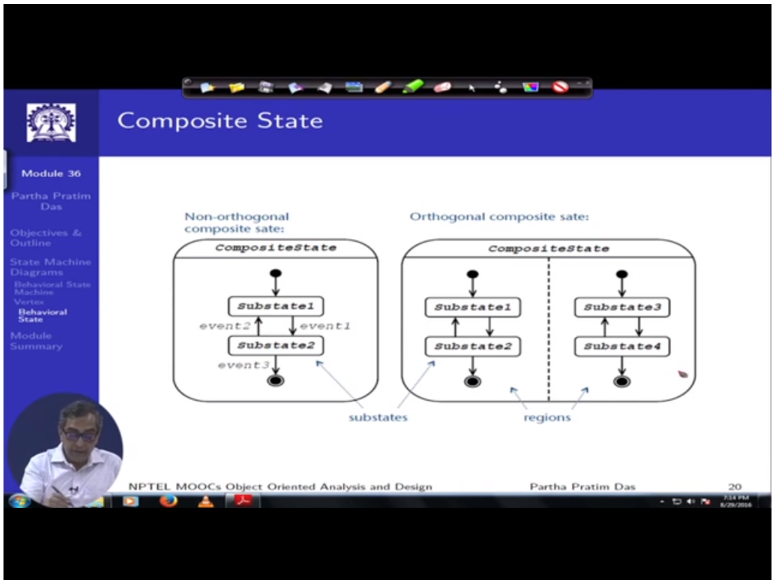
And these sub states can happen in a sequential manner in a disjointed manner like it is happening here or it could happen in a concurrent or orthogonal manner. it could happen in terms of what is known as different regions, a region is a in a composite state is a collection of states and transitions that define one state machine in that part which is giving you that composite state and in terms of these are composites and composites of composites and so on, you can very compactly represent a state machine diagrams.

(Refer Slide Time: 23:14)



Now composite states could be orthogonal, they are called orthogonal if there are there has more than one region that is more than one independent traits of execution that happen in the composite state and when we have that we can actually show the sub states in terms of just a symbol instead of showing the details. in the earlier slides we show they showed the details of serving customer here we just showing as a icon, meaning that there are sub states in this which describes the details of the serving customer.

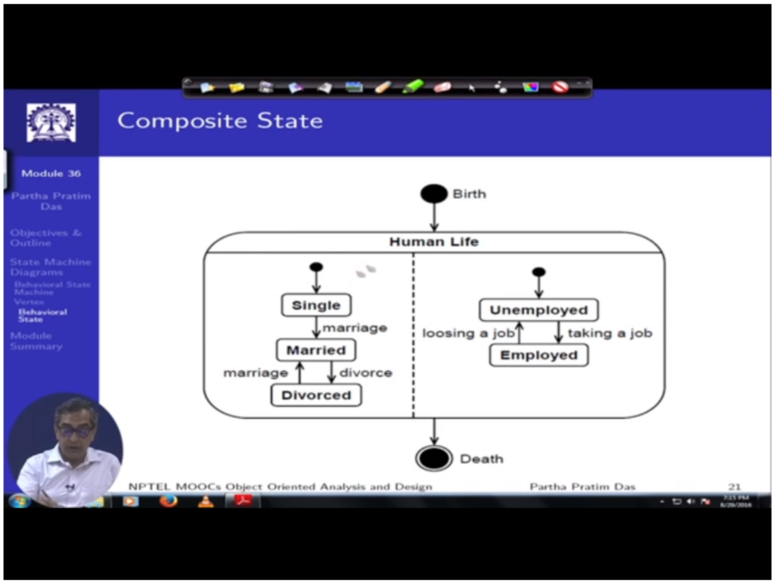
(Refer Slide Time: 23:48)



So this is kind of the composite state you might have. This is non orthogonal composite state which means that this has only one region. So this is only one state of execution, this is where you start and these are your states, on certain event 1 you change from sub state 1 to sub state 2, on event 2 you change back and on event 3 you comeback. Whereas in an in another composite state which is called orthogonal you will have more than one region.

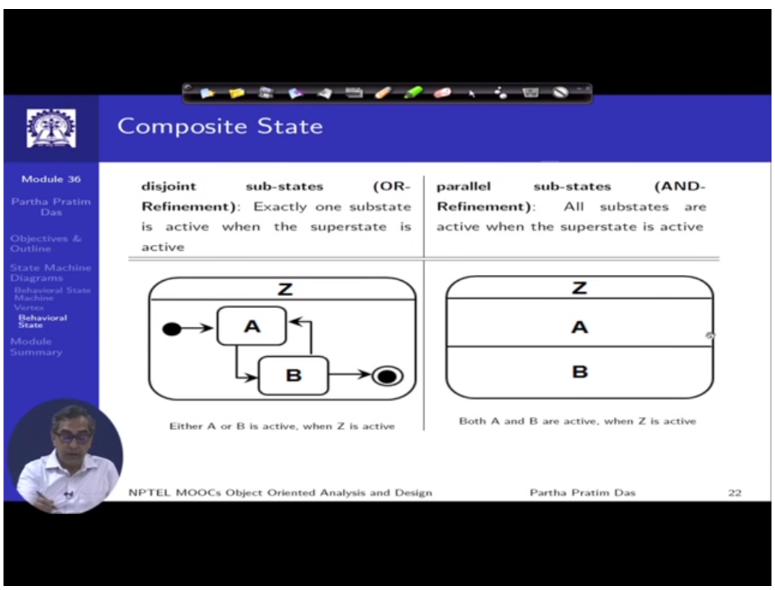
So this is one region, this is another region, these are the 2 different regions and they are typically shown by separate dotted lines and you have sub stakes in region 1, you have sub states in region 2 and you have regions transitions happening between them but these are independent, these are kind of this what is happening here is in concurrence with what is happening here. So in time domain their happening is has concurrent events and when that happens then you says that is an orthogonal composite state.

(Refer Slide Time: 24:51)



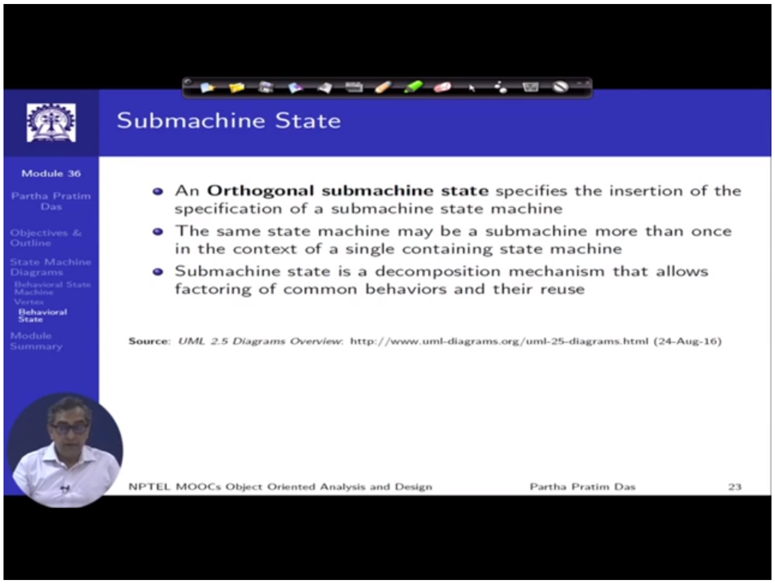
So this is one example of a human life, so these are these are 2 orthogonal 2 regions, 1 region of personal life and 1 is region of professional life and you have multiple different transition states happening between the sub states of the composite state of the human life single, married, divorced, whereas on the other one unemployed, employed. So these every region will have certain different state of which will act independently on the overall composite object.

(Refer Slide Time: 25:23)



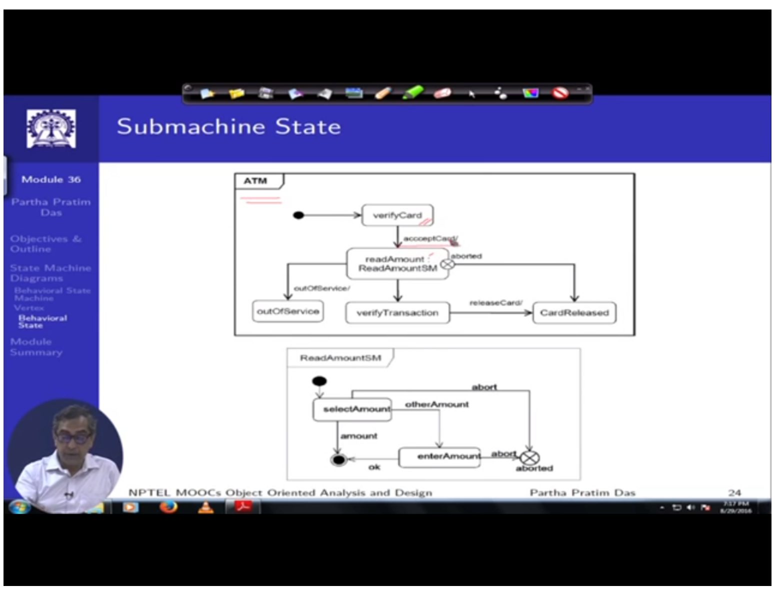
So you could show these in terms of disjointed sub states as in here or you could show them as parallel sub states depending on if they are shown like this they are disjointed, if they are shown as parallel, so they are like different regions they happened concurrently. So these are the different notations that you need there.

(Refer Slide Time: 25:43)



Finally a third kind of a so in terms of a behavioral state, we have simple state, we have seen composite state; we have seen simple composite state which has only one region, a non-orthogonal composite state and we have seen orthogonal composite state. Another that is commonly used in the representation is called submachine states. So you can define sub machines of a machine and actually define properties based on that. So submachines so let us try to look at an example that is easier to understand.

(Refer Slide Time: 26:22)



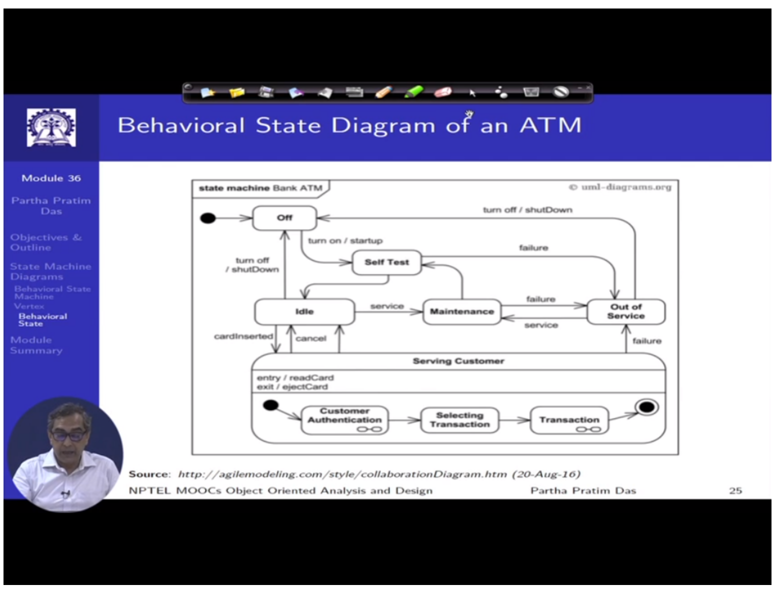
Say we look at atm itself, so the states are you verify card and once that is accepted, the card is verified then you read the amount and do the transaction and the outcome could be that the transaction is aborted where you moved move to card released there may something goes wrong we will go to out of service or you may actually see if the transaction is verified and then the card is released by part. So this reading amount and actually you know doing the all those balance check.

And all that is shown as one state but is actually another state machine itself. So the we are we mark this as read amount sm am just writing sm, we get that sm is submachine, so read amount sm we will expect that there will be another state machine diagram which gives the details which says how to select the so when you are accepting accepted the card then you are here and you have select the amount and you have checked the amount is okay, you enter the amount and all that.

And at the end you could either remain reach the final state which is success state which is here or you could you reach the abort state which is here. So sub machines are very frequently used so that a so if they are like the sub machines are like function calls, they are like subroutine calls, so we are saying that here I have a submachine and this same submachine could be used in multiple other finit state machine diagrams as well behavioral state machine diagrams as well.

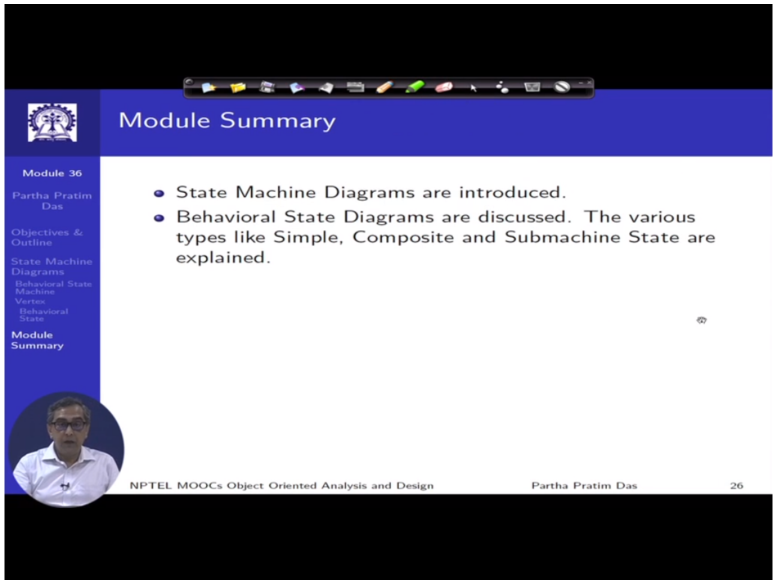
And they all will by the name refer to the same submachine where you are. So the submachine state or states which basically have complete submachine diagram incorporated in terms of it.

(Refer Slide Time: 28:10)



So given this is the behavioral state diagram of an atm. I will not go through the details, we have just briefly discus this and I will request that you please go through each of these state transitions and get comfortable in terms of describing an atm system to this kind of an fsm.

(Refer Slide Time: 28:31)



So to summarize state machine diagrams had been introduced, we have talked about what a state machine diagrams are and we have given an overview in terms of a behavioral state machine diagrams particularly talking about vertex and the different behavioral states, the simple states, the composite states and the submachine state. In the next module we will continue discussing about other kinds of states in a state machine diagram.