

Object – Oriented System Development Using UML, JAVA and Patterns
Professor Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture No 22
Example of State Machine Modelling

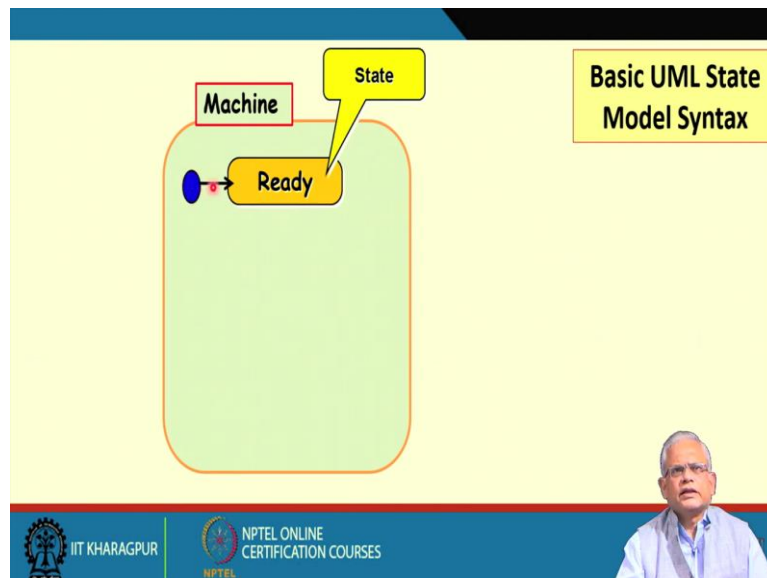
Welcome to this lecture.

In the last lecture, we had looked at some features of UML state machine, which distinguish it from the FSM. UML state machine model is a very powerful mechanism compared to the typical FSM. We looked at some features, we looked at the nested state and the concurrent states, which are called as the AND states. Both of them constitute the composite state. We see also state machines enclosed within the state that is nested state.

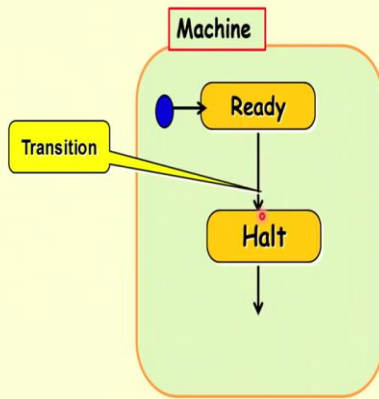
Now, let's proceed from that point onward. Now, we are still looking at the basic syntax. And as soon as we complete looking at the basic syntax of the UML state machine, we will proceed to model some simple problems. We will construct the state machine representation from those simple problems. And then we will see how to generate code.

Once we have the state machine model drawn, it's a trivial thing to generate the code. We will see that if we understand the methodology, the code generation is easy. The case tools which support state machine model, they can generate code based on clicker switch press. Now, let's proceed from what we are discussing.

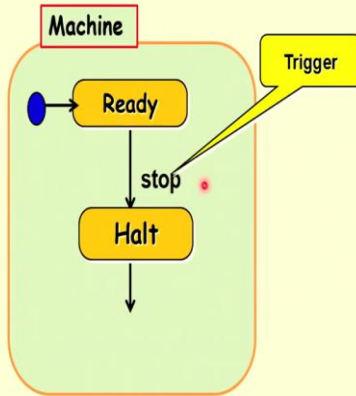
(Refer Slide Time: 2:07)



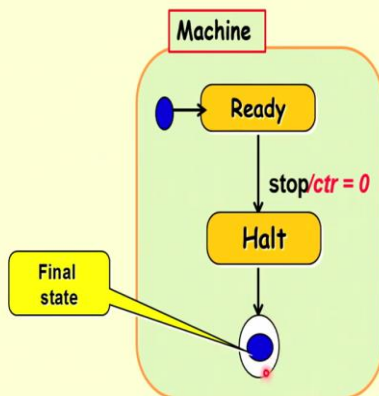
Basic UML State Model Syntax



Basic UML State Model Syntax



Basic UML State Model Syntax



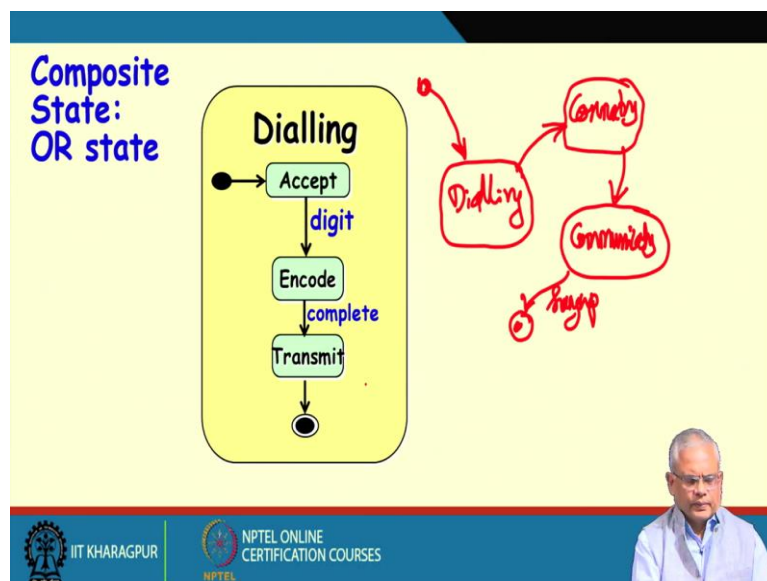
We are discussing about a composite state. The composite state can be two types, the nested state and the AND state or the concurrent state. We might name a concurrent state in this way (in the above slide). We write that this is the state of a machine or we might also write the machine inside the state here. Both are accepted. And then we find that inside the machine state, this is a composite state it embeds another state machine. That is why it is called a composite state.

As it enters the machine state, it has, the pseudo state indicates that it starts with the ready state. And then we have the transition. In the transition, we write the event here, which causes the transition, there may be guards associated, which are conditions that must occur and the event must occur for the transition to take place, goes to another state Halt when the stop switch pressed. So, that we call as the trigger or the event.

We might have guards associated and also some actions associated. The action, we write with a slash here. For example, here, the action is the variable because this is extended state machine and there is a state variable control, which is initialized to 0.

As it goes from Ready to Halt state on the stop switch press, the control state variable is set to zero (ctr=0). And then from the halt state it goes to the final state. The final state is written as a filled circle, which is enclosed within another circle (as shown in the above slide). So, this is the basic UML state model syntax.

(Refer Slide Time: 4:53)



Now, this is a composite state: OR composite state (in the above slide). In the OR state, it can be any one of the states, Accept state or Encode state or Transmit state. The name of the state

is Dialling. The telephone is in the Dialling state. It is waiting for the key press. It accepts the key or the digit encodes the key. And as the encoding is complete, it transmits.

To start with, it is accepting the digits. And then it encodes the digits. As the encoding is complete, it transmits the digit. So, that's the representation of an OR state. We might also represent this as just dialling in a top-level diagram and then it is connecting and then it goes to the state communication. The parties are communicating and then hang up event. So, we can represent this as dialling, connecting, communicating, and then hang up.

And then we might represent the dialling state as another diagram. In a case tool, if we click on the dialling state, it will show this composite state dialling state. It embeds another state machine. But then the composite state can also be an AND state. The OR state is any one of the states, it can be present. It can also be an AND state.

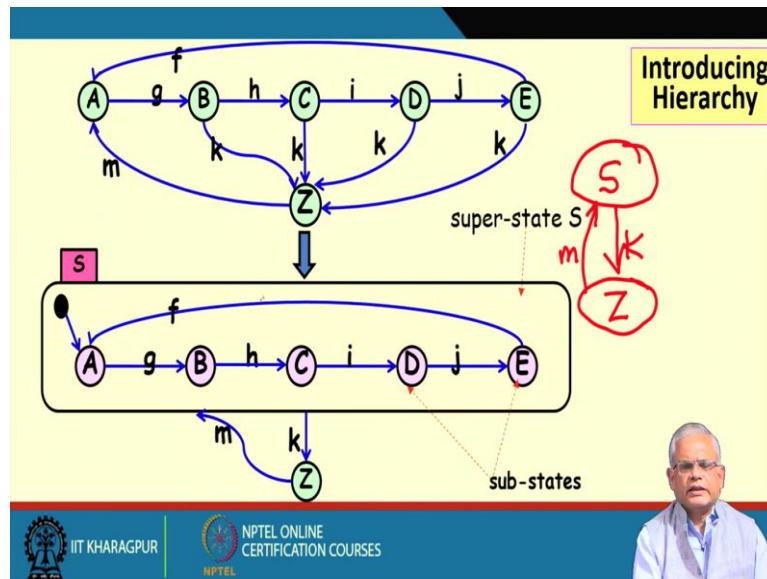
(Refer Slide Time: 8:01)

The slide features a yellow background with a blue header and footer. A yellow box in the top right corner contains the text "Composite States: AND/OR States". The main content includes a bulleted list and a state machine diagram. The diagram, titled "Robot Controller", shows four states: "Walk", "Forward", "Run", and "Backward". "Walk" and "Forward" are at the top, separated by a vertical dotted line. "Run" and "Backward" are at the bottom. Transitions are shown with arrows: "r" from Walk to Run, "w" from Run to Walk, "b" from Forward to Backward, and "f" from Backward to Forward. A small "OR" label is next to the Backward state.

- Composite states can have:
 - **concurrent substates -and- relationship**
 - substate separated from others by dotted line
 - **disjoint substates -or- relationship**
 - transitions between substates

The concurrent substates, we call this as the AND relationship between the states and the disjoint substates is the OR relation. Let us say, this is the robot controller state (in the above slide). To start with, it is walking and in the forward direction, but if we press the run key on the remote, it goes to the run state. And therefore, the robot will be forward running. If we press the back, it will be backward running. If we press the walk, it will be backward walking and so on. So, depending on the switch setting, it will be in one of this state machine. So, that is a AND relation represented by a dotted line.

(Refer Slide Time: 9:14)



Now, let see, how do we really construct or convert a traditional state machine into a composite state. Let say, we have a state machine like this (in the above slide), that there are transitions occurring between these states A, B, C, D, and E. And then there is this m transition is taking in A and k transition is coming out. Looks a bit complicated but FSMs normally look like this, lot of states and transitions, even for very simple systems.




If we want to represent it in the form of a composite state in a state machine diagram, we will observe here, that see from each of this A, B, C, D, and E on k it is coming out to Z state. And therefore, if we represent A, B, C, D, and E in one super-state, just one arrow is enough from the superstate and to the superstate. This arrow from the body of the super-state to another state indicates that on k from any of these states, it can come to the Z state. And to represent that on m, it goes to A state means it comes to the body of the super-state. And then we have the pseudo transition to A indicates that if from Z on m event it comes to the S state or the super-state. We call these A, B, C, D, and E as the sub-states. And this whole thing is the super-state.

Sometimes, we might just show the details of the super state in a different diagram. And that's how, this diagram will represent the first level diagram very simply by drawing the state S, which is a composite state. And then it transits to the Z state on k and goes to S state back on m. This state machine, at the first level we draw this diagram and S is a composite state. And we can represent the S state in another separate diagram, using this state diagram (in the above slide, hand draw diagram).

(Refer Slide Time: 13:02)

Exercise

- A printing machine is in the power off state to start with.
- When powered on, it goes into a startup state.
- After completion of the startup state, it enters operational state.
- Whenever, a problem is detected, it enters an alarm state.
- Whenever it is switched off, it goes into a power off state.

Now, we have seen the basic syntax so far. Can we use this basic syntax we learned to develop the state machine diagram for simple problems?

Let say, a printing machine is in the power off state to start with. A very simple problem. And therefore, we have explicitly indicated the states here by reading. We can easily identify the states, which we can represent in the form of a diagram.

The printing machine is in the power off state to start with that means the pseudo transition to the power off state. And then, when the power switch is pressed on, it goes to the startup state. After completion of the startup state, it enters the operational state. However, whenever a problem is detected that is either it is in the startup state or in operational state, it enters an alarm state. So, the states are clearly mentioned here.

We have the states clearly mentioned here. Sometimes in the text description of the behaviour of a system, the states may not be explicit. We might have to apply our thought and identify the state machine. But here things are simple. We have this power off state, the startup state, operational state and the alarm state.

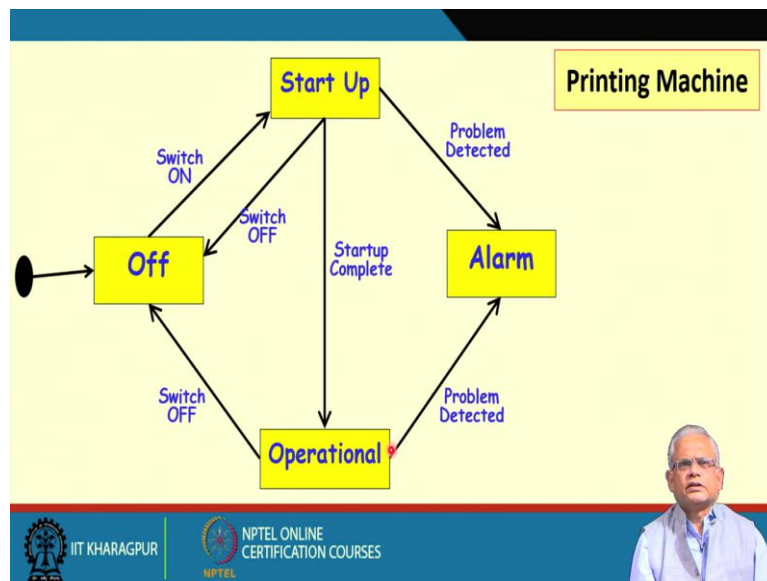
The printing machine is in the power off state to start with. And therefore, we draw a pseudo transition to say that this is the state to start with. When powered on it goes into the startup state. After completion of the startup, it enters the operational state. But then, anytime a problem is detected either during the startup or the operational, it goes to the alarm state. So, we will draw that the problem event occurs (in the above slide). When switched off, it goes into a power off state. In any of these states again, whether it is operational, startup, alarm, if

it is power off, it goes to the power off state. The diagram shown in the above slide with all transition and events.

From the text description, it is not hard to construct the state machine model, but this is not the best model actually, because just observe that there are some of the transitions from states are common. See here, the problem event as it occurs, it goes to Alarm state. That indicates to us that this can be a super-state, that will reduce the number of transitions occurring from the state. We will just have one arrow, problem arrow coming out of the start and operational state. And similarly, the off is between the startup and the operational state. That will really simplify this state machine, help it to be a state machine representation which will become easy to understand.

Let us first draw this diagram little nicely, and then we will see how to have this composite state introduced to be able to draw a very simple and tidy diagram.

(Refer Slide Time: 18:39)



So, this is the same diagram we drew there by hand (in the above slide). The startup state, alarm state and operational state. By default, it is in the off state as the printing machine gets started. It goes to the Start Up state and as the Start Up is complete, goes to the Operational state. But in Start Up or Operational, in any of this state, if the problem is detected, then it goes to an Alarm state. And in both the states, Start Up and Operational, if switch off, then it goes to the off state.

Both switch off, and Problem Detected events are recognized in these two states. If the switch off occurs either when it is in Start Up or Operational, it goes to the off state. This is the initial state machine diagram, we can easily construct but then, we need to introduce the composite states which will help this diagram to be organized well, more understandable and tidy. And for that we notice here that the switch off event is common between the Start Up and Operational, both of them go to the Off state. And similarly, the problem detected here, these two states behave very similarly.

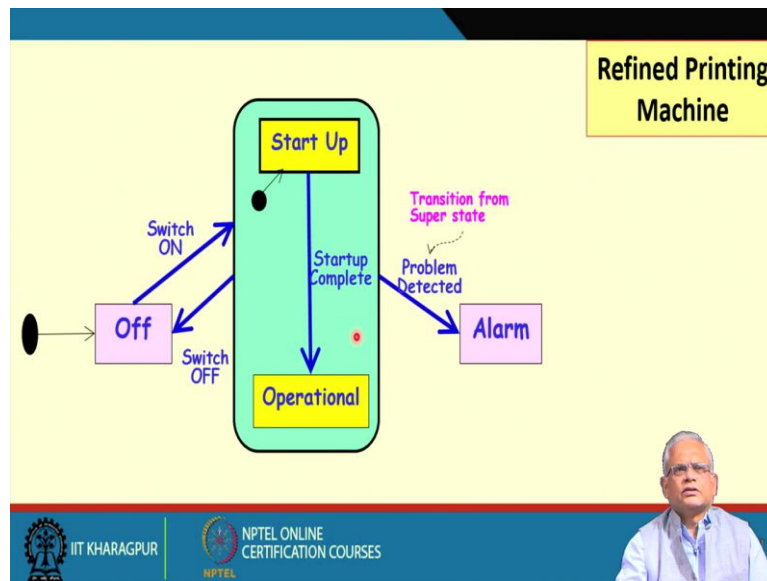
So, if we put these two states in a composite state then our diagram will simplify because we will have just one arrow going on switch off to the off state and another arrow going from the super-state to the alarm state on problem detected. The number of transitions will increase, the diagram will appear tidy. And we would have introduced hierarchy into our diagram, which will make it easier to understand.

And let me just repeat that whenever we see that from different states, the transition to occurring another state with the same event, we would might as well make the states from

which the transition occurring into a super-state. You might ask that, is it possible to do it automatically?

No, from our diagrams, it should be possible to do automatically, but then for our purpose, we will check it manually. And then we will redraw the diagram. Now let's redraw this diagram introducing the composite state and making it a nested diagram.

(Refer Slide Time: 22:06)



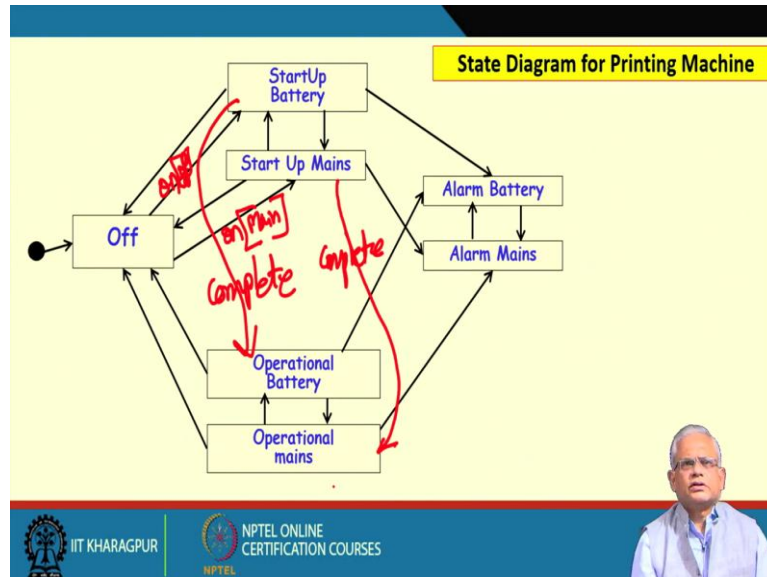
So, this is the nested diagram (in the above slide), we had noticed that the switch off takes it off state as the printing machine to start with is in the off state. On switch on, it transmits to the on state. Maybe I should have written here, the on state (the big round rectangle in the above slide), in the on state, as the switch on, it starts at the Start Up state, that is indicated by this pseudo start state and transition here. As the Start Up is completed, it goes to the operational state. And in either of these two states, if a problem is detected then it goes to the alarm state.

So here, just see that the arrow is coming out from the super-state that means from any of the sub-states, this transition can occur. Whenever the problem detected event occurs, when the system, the printing machine is any of its state, it will go to the alarm state. And similarly, in any of the on states of the printing machine like the Start Up and the Operational, if the switch off event occurs, then it goes to the off state.

And typically, we will represent the top level as a simple diagram. We will write off and then on state. In the on state, we will have switch off and switch on. And then problem detected. And the on state, we might draw a separate diagram, which will show the state machine, and

therefore, we understand the top-level behaviour and then we look at other behaviours. And of course, in a case tool, we just draw the off state, on state, and alarm. And if we double click on the on state, then this state machine will show (the Start Up and Operational states).

(Refer Slide Time: 24:27)



Now, let say, we have a situation that in the printing machine, we have battery backup. We have additional features of the printing machine. To start with, it is in the off state and let say, as we press the on switch the main power is on, it goes into the Start Up mains, using the main power supply. I have not written the transition events here because that would clutter the diagram. And right now, our focus is not to get the complete diagram, but just to see the features. If main switch is on and the on switch is pressed, it goes to the Start Up Main and if the main power is off and we press on, it goes to the StartUp Battery. And any time we press the off, it goes to the off state. And if the power comes back, when it is on the battery, it starts running Start Up, using the main. And if the power goes off at any time, it goes to the StartUp battery. And anytime alarm occurs, it goes to the alarm using battery or alarm using main. And as the startup completes, I have not even drawn the arrows here because that will clutter the diagram, there will be more arrows actually. As the startup completes, it goes to the startup main and from startup battery, it goes to the Operational Battery.

Now, can we draw this diagram using the composite state? Because looks complicated with lots of transitions. Even I have not written the labels here and omitted some of the transitions here, but still looks complicated. Can we redraw this diagram using the composite states? Please try yourself.

We will see the answer in the next lecture. Please try drawing this diagram and remember that there are some other transitions here. One is that from the startup main as the startup is complete, it goes to the operational main. And from Startup Battery, it goes to Operational Battery if the startup is complete. And the on switch is pressed when the main power is available, it goes to start up main and so on.

Please try this exercise. We will see the solution in the next lecture.

Thank you.