

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

Lecture – 45
Activity Diagrams: Part II

Welcome to module 33 of object-oriented analysis and design.

(Refer Slide Time: 00:30)

mod33lec45

Module Objectives

- Understanding Control features of Activity Diagrams

Module 33
Partha Pratim Das

Objectives & Outline

Controls
Initial Flow
Final Activity
Final Node
Decision Node
Merge Node
Fork Node
Join Node

Example
Summary

NPTEL MOOCs Object Oriented Design and Analysis Partha Pratim Das

We have been discussing about activity diagrams and we have introduced the basic notions for it. So, in this module our objective would be to specifically look at the control features of activity diagrams.

(Refer Slide Time: 00:42)

Module Outline

- What are Activity Diagrams?
 - Activity
 - Partition
 - Activity Edge
 - Control
 - Objects
 - Actions
- Activity Diagram for LMS

Module 33
Partha Pratim Das

Objectives & Outline

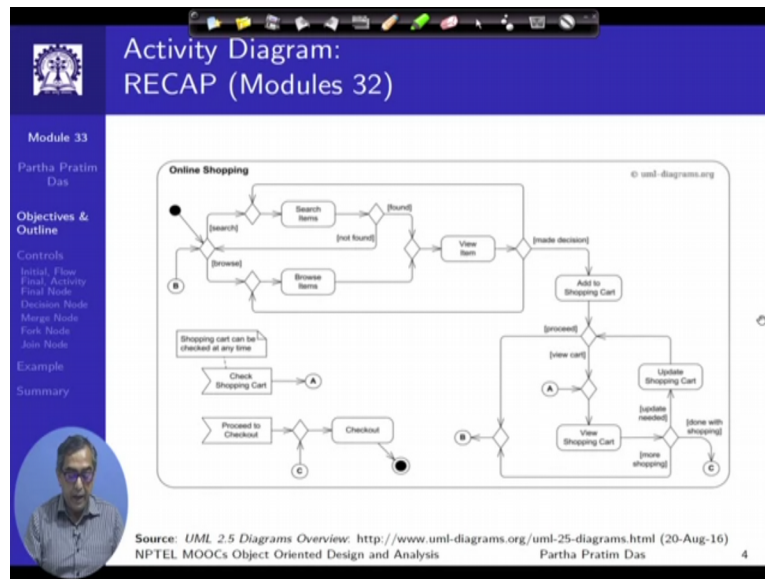
Controls
Initial Flow
Final Activity
Final Node
Decision Node
Merge Node
Fork Node
Join Node

Example
Summary

NPTEL MOOCs Object Oriented Design and Analysis Partha Pratim Das

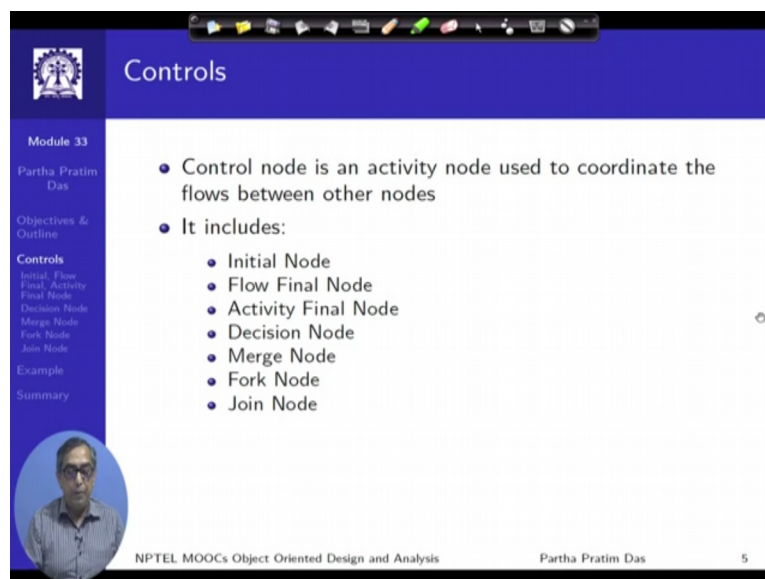
The outline as I had mentioned in the last module comprises this whole set that define the activity diagram and control which is shown in blue is specifically what we covered here.

(Refer Slide Time: 00:52)




So, we would quickly recap, we saw the activity diagram defining activities, the partitions and ages including the object flow ages and we looked at guards and all those other related points and we specifically discussed about this particular example of online shopping.

(Refer Slide Time: 01:26)



So, before you go through this module I will request that you would revise on that online shopping exercise that we have discussed at length. Now in terms of control besides while certainly activity diagrams have activity node and have some other nodes but significantly they have a whole lot of control nodes. So, we have defined the edges between activities but we will see that actually there could be edges between variety of different nodes in this graph kind of structure.

(Refer Slide Time: 02:11)



Module 33
Partha Pratim Das

Objectives & Outline

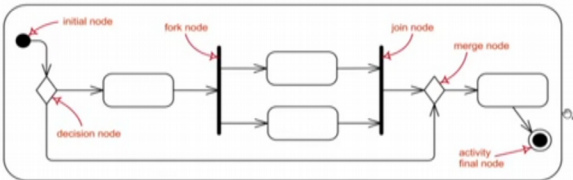
Controls

- Initial Flow
- Final Activity
- Final Node
- Decision Node
- Merge Node
- Fork Node
- Join Node

Example


Summary

Activity Control Node Overview



Activity control nodes overview

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



NPTEL MOOCs Object Oriented Design and Analysis

Partha Pratim Das 6

So, these are the different kinds of nodes that are possible in terms of the control behavior, in terms of the control node. So, we will try to understand what they mean before I get into the exact definition of each and every node, let me just first give you an overview in terms of the symbols that are involved. So, we will explain each one with the behavior subsequently. So, in terms of the symbol let me use blue. So, this as we have seen a solid glob is an initial node, that's the beginning of an activity.

So, this is an initial node and if that is circled within another mt1 then it's an activity final node. So, the whole activity is a flow from the initial node to the final node. We have seen that these are are different activity nodes, these are the different activities. So, an activity will have sub activity, a sub activity may have sub sub activity and so on depends on the granularity of how much depth you want to go in. next we will see control nodes which are decision nodes.

A decision node basically we will take a decision and will have at least 2 outgoing edges based on the outcome of the decision. The decision could be binary, it could be so here it is a decision node which shows one possible outcome and another possible outcome. Then there are merge nodes which are shown by the same symbol of a blank diamond where more than one edges actually come and then one single edge go out which show that how the different flow that was coming from this side and that was coming from this side actually flowing to the next one.

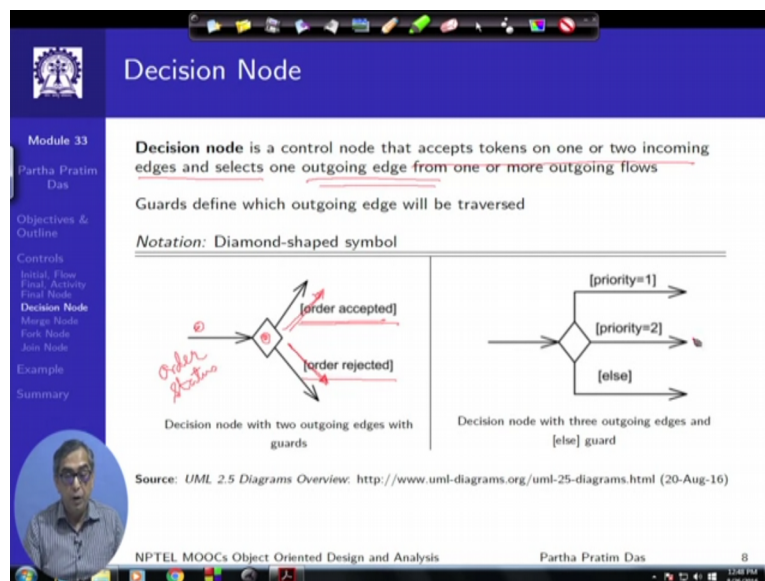
So, this is the merge node, so if you look into these components then this is pretty much like the flow chart notation that we are using. What is new are these 2, the fork node and the join node, the fork node basically is introduced for concurrency. So, the basic meaning of fork node would be that the flow is coming from this edge through this edge after this activity has been done and then

concurrently both these edges actually activate that is both these activities will start at the same time is a concurrent operation.

On the other hand, a join node waits for the different flow that had started concurrently to reach a common point of completion and then only this will proceed. So, this is known as a join node. So, these are very typical of activity diagrams which show you the concurrent part of the algorithmic behavior or the use case behavior where more than one activities are happening at the same time which is very very common in most situations.

Those of you who have done some course on operating system would be familiar with this notion of concurrent behavior, you would also be familiar with the fork and join operations that typically you see in Unix processes and so on. So, these representations basically those kind of fork, join behavior where concurrency can be modeled. So, this is the basic overview of activity control nodes and how those controls will look like.

(Refer Slide Time: 05:40)



So, if we go into depth the initial node is a control node, let me revert back to red, initial node is a control node which where starts the flow starts at this point. It is shown with a small solid circle as we have seen. Final node is one with a small circle with a cross inside. So, this there is a difference between we have already seen this which is an activity final node, this is a control final node that stops all flows in an activity which we have already seen, this notation.

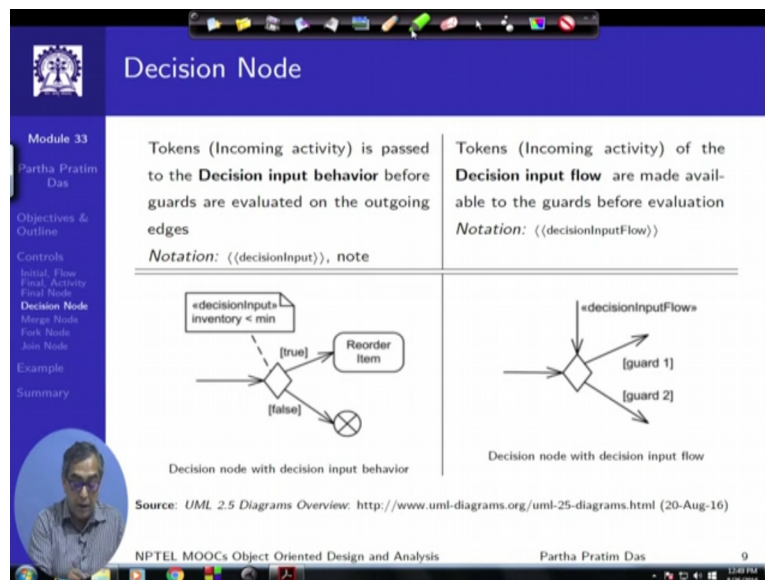
But there is a flow final node also which basically say that this not only this activity has completed but it is a total flow stops here. So, there is nothing further to happen on this. So, you put a flow final node for this, in some diagrams you will find this notation happening.

Moving on these are decision nodes as mentioned that accepts a token from one or two incoming edges, there could be more also and there and selects from the outgoing edge. So, there is a token coming from here, some token coming from here that is some information coming through this and this decision node based on that token decides either to take the control this way or to take the control this way and typically you show guard conditions here showing what is the value that the token has to take so that the decision is made.

So here there is some order status so the token coming in here is some kind of an order status and if that has a value, order accepted you take this flow, if this order rejected, then you will take this flow which is typical of in in flow chart we typically just have if kind of binary decision, we just say true, false or yes, no but here it could be multi way. So, it is important that you specify the guard conditions there.

For example, we are showing it here, this is where you go if priority is 1, this is where you go if priority is 2, this is where you go if priority is something other than 1 and 2. So if the token coming in here is a priority value and based on that a decision, a multi-way branching is made.

(Refer Slide Time: 08:09)



So, these are the decision nodes decision nodes could have decision input behavior or decision input flow. The decision input behavior says that you can this is marked in terms of a keyword like this, just note how the keyword is. If you you must have noticed that in uml very regularly we put things within this. So, whatever you put in within this corner brackets, double corner brackets are basically keywords of uml.

So, this is this says this is a decision input, so that decision input is nothing but the condition that that particular decision node has to satisfy. So, if you look at the decision input here is inventory is less than minimum that is here what comes in is the inventory. So, prior to this there must have been some activity which has changed the amount of inventory all as affected the amount of inventory and then in this is this decision node, your decision input is if inventory is less than certain minimum value and your guard conditions are true or false.

So, if you come here, then it is true which means that the item can be reordered was in this case say and if it is false then you get get a flow final termination that is in this flow, there is nothing else that need to be done, pursuit. In contrast, there could be input decision flow or incoming activity, so it is input decision flow. so where the ba condition basically comes from this decision flow, here the condition is fixed by the decision input, here the condition also comes from the decision input from this edge.

And based on that ehe condition the different values of those condition you make this branching. So, this is little bit more generic in nature but so if it not known as to what could be the different values or what could be the different conditions on which you branch, you may need to design based on a decision input flow, otherwise you design based on decision inputs.

(Refer Slide Time: 10:18)

Merge Node

- As a **Merge node** is a control node that brings together multiple incoming alternate flows to accept single outgoing flows (Control and Object Flows)
- Merge Node is non-blocking**
- Notation:** Diamond-shaped symbol with two or more edges entering it and a single activity edge leaving it

Merge node with three incoming edges and a single outgoing edge

Merge node and decision node combined using the same symbol

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)

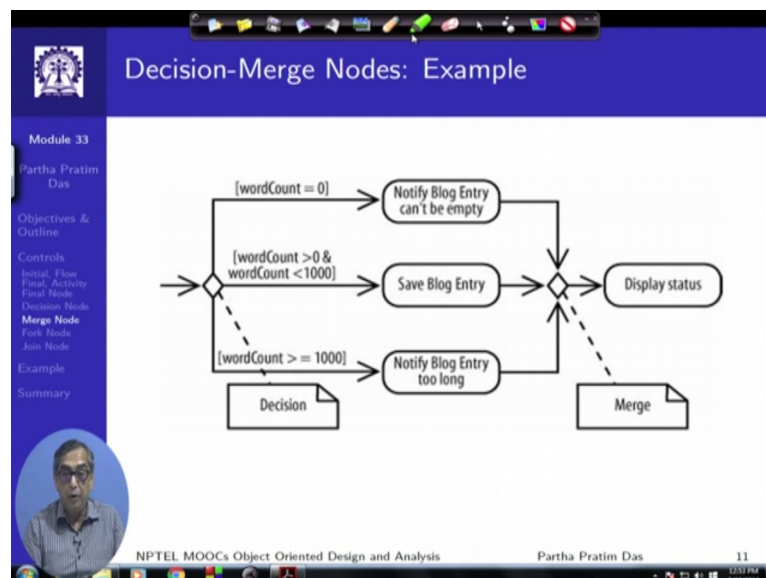
NPTEL MOOCs Object Oriented Design and Analysis Partha Pratim Das 10

Similarly, you have merge nodes. The merge node is a control node again and brings together multiple incoming alternates flow into to accept a single outgoing flow. So, there are as you can see the symbol is same, so there are multiple ways through which the control can come here and whichever way it comes, it flows out from here. It is non-blocking, so even though there are multiple inputs it just the flow continues as soon as any one of them has arrived.

So, if there is a token here that immediately gets transferred here that is the flow continues in this manner alternately if something first something comes here then the flow will continue like this. So, this is where and certainly it will have only one outgoing edge, so to make it a merge node. But you can combine the decision and merge node in terms of a structure like this were here 3 flows are merging in this node. At the same time, there is some decision input based on which the flow is branching out in 3 different ways.

So if it is that you have a unique condition through which this branching can be decided and this is a point where so other way another way to show this could have been that you do this, you show this, this, this as a merge, then you have a outflow and then you have possibly a bank activity, this is nothing that needs to be done or actually you could just let us let us do it differently, the flow is coming in here, the flow is coming in here, it goes out and you have another diamond,

(Refer Slide Time: 12:24)



So, this is the merge node, this is the decision node and do these 2 together could be combined and shown in a combined fashion. So, this is a merge decision node. So, these are some examples, for example here we are trying to just refer to the blogging example we had started off in the last module, so we are blogging, so if so some blog has been created. So, in if the word count in the blog is 0, that is a good condition that then the activity is notified blog entry can't be empty,

Certainly, you cannot have an empty blog. But if it is greater than 0 but less than 1000, this is a guard condition then you save the blog entry if it is more than 1000, you notify again then the blog is too long and it cannot be put up. So, these activities are or different and based on this condition 3 conditions a decision is made at this point. But whatever that the decision is, you finally merge them all together, merge all the flows together and display the status.

So, this activity is common irrespective of whether you have gone through this or you have gone through this or you have gone through this. The typical flow chart logic.

(Refer Slide Time: 13:24)

Concurrency: RECAP (Module 10)

- **Concurrency** is *the property that distinguishes an active object from one that is not active*
- Allows multiple tasks to execute, interact and collaborate at the same time to achieve the global functionality
- Concurrency is critical for *Client-Server Model* of computation

Module 33
Partha Pratim Das
Objectives & Outline
Controls
Initial Flow
Final Activity
Final Node
Decision Node
Merge Node
Fork Node
Join Node
Example
Summary

NPTEL MOOCs Object Oriented Design and Analysis Partha Pratim Das 12

Concurrency we have discussed about that so now we will just again discuss in depth about the concurrency node. So, I have just explained it at the beginning.

(Refer Slide Time: 13:35)

Fork Node

- **Fork node** is a control node that has one incoming edge and multiple outgoing edges and is used to split incoming flow into multiple **concurrent** flows
- **Notation:** Line segment with a single activity edge entering it, and two or more edges leaving it

Fork node with a single activity edge entering it, and three edges leaving it

Combined join node and fork node


Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)

Module 33
Partha Pratim Das
Objectives & Outline
Controls
Initial Flow
Final Activity
Final Node
Decision Node
Merge Node
Fork Node
Join Node
Example
Summary

NPTEL MOOCs Object Oriented Design and Analysis Partha Pratim Das 13

So, fork is the first kind of control node for concurrency which has one incoming edge that is activity that was going on is flowing it through this edge and it has multiple outgoing edges. So, it has to have at least 2 to mean concurrency that is used to split incoming flow into multiple concurrent flows. So, this is a sequential flow and these are all concurrent flows. So that is what you have to keep in mind that once the control reaches here, this token reaches here actually as if 3 tokens, 3 separate control start happening at the same time.

(Refer Slide Time: 14:20)



Join Node

Module 33

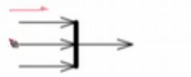
Partha Pratim Das

Objectives & Outline


Controls

- Initial Flow
- Final Activity
- Final Node
- Decision Node
- Merge Node
- Fork Node
- Join Node**
- Example
- Summary

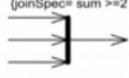
- Join node is a control node that has multiple incoming edges and one outgoing edge and is used to synchronize incoming **concurrent** flows
- Join Node is blocking
- Notation: line segment with several activity edges entering it, and only one edge leaving it



Join node with three activity edges entering it, and a single edge leaving it



Combined join node and fork node



(joinSpec= sum >=2)

Join node with join specification shown in curly braces

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)

NPTEL MOOCs Object Oriented Design and Analysis

Partha Pratim Das

14

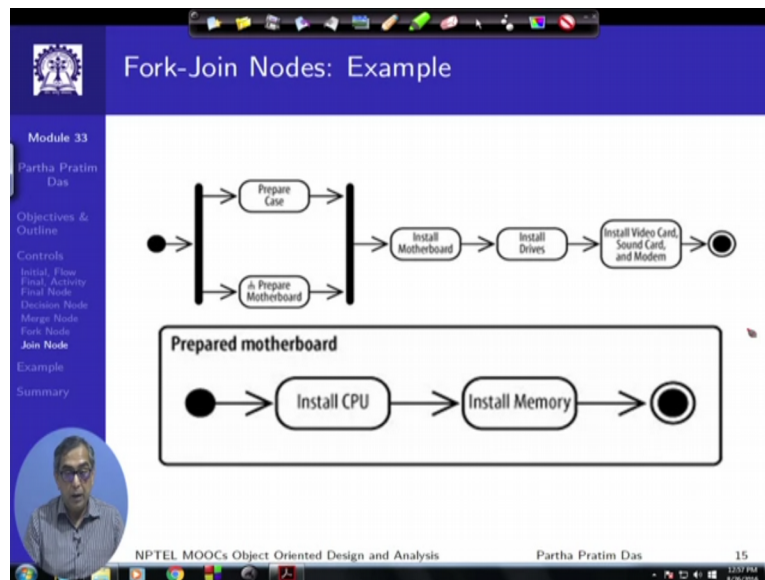
So that is a fork node which is clear and then you will have the other side which is joined. So here you have 3 controls coming in and these 3 controls when you put them in the join node, then the basic difference is this is combining through blocking which means that if a token comes through this that is if the control the prior activity to this is over, completed so the control comes here, the control will not start from here.

It will not be able to start it it will have to wait till the token comes here and the token comes here all 3 tokens arrive, that is a blocking part of it, that is a critical. So, you will have to understand the crucial difference between a merge node which is this and the join node. In the merge node, this is the merge node if the token comes from any one of this, that is if any one of these flow has come through any one of these edges, edge 1, edge 2 or edge 3, the flow will immediately start on the outbound edge.

But here if the flow comes from one it will have to check whether the flow has completed on 2 or 3 only when all these 3 are completed then the token will go there then that will. So, if we say this is kind of a condition on control flow, this is kind of an and condition and control flow certainly this is related to concurrency because by fork we have made that from one single flow we have created 2 or more flows happening concurrently. So, we we have to recombine them back into a single flow through the join node.

So that is the important part and as you can merge decision and merge node you can also merge the combine the join and fork node together. So, it is you you fork here there are some activities which must be obvious from the context and then then you wait for all of them to complete together.

(Refer Slide Time: 16:34)



So, we just am just showing an example here this is a as if you have to define the activity, define the process for preparing a say a desktop computer to be dispatched. So, you start here and you fork because preparing the computer means preparing the case within which it is housed and preparing the motherboard and possibly 2 different teams work on this. So, they need not happen sequentially one after the other but they happen concurrently.

So as soon as the indication come that the computer has to be prepared, the prepare case as well as the prepare motherboard activity start and only when both of them are over, certainly you cannot install the mother board you you see what is the next activity, next activity is installing the motherboard. So, you cannot install the motherboard unless the motherboard is ready and also the case is ready. So, both will have to be ready.

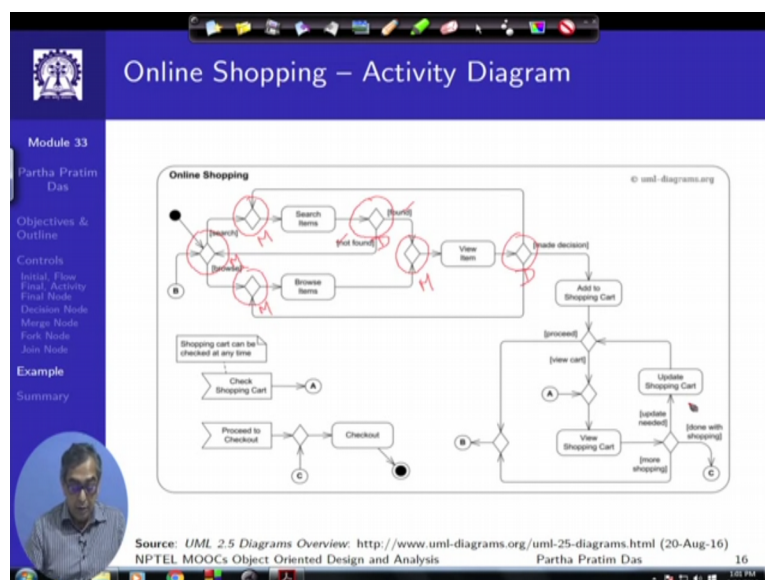
So, this is a that is why here you have to fork because independent activities have to happen at the same time and here you have to join because the 2 activities that were happening independently we both have to complete and only then the following activity can happen and rest of the activities follow sequentially, install the motherboard you install the drive, installed video cards exedra and you are done with the final of the activity so that is the basic purpose of doing fork and join

And you can also see here we are I am using something which is a very special symbol, a three-way this inverted three-way symbol here, just note that this one, this means that the prepared motherboard activity is actually another more complex activity and this have been defined somewhere else were more detailed activity diagram. So, it is like I mean conceptually if we think in terms of programming languages this is like a function call kind of your making.

So, you are saying that this is detailed further so we will look for where is the prepare for motherboard activity and there is a prepare motherboard activity. So, this actually means this whole activity so in place of having just a single activity you have this whole activity of again you have a start for this activity of an activity final node, so you install CPU, install memory and complete the prepare motherboard activity.

So, this is often how you show a more complex activity as a part of some other activity and certainly this enhances the reuse in terms of the activity diagram because this may not be the only place where you need to prepare motherboard, you may need to prepare motherboard in terms of a repair operation also, repair process also and you will be able to simply reuse this activity to do that. So, this is how you make use of the fork and join to decide on.

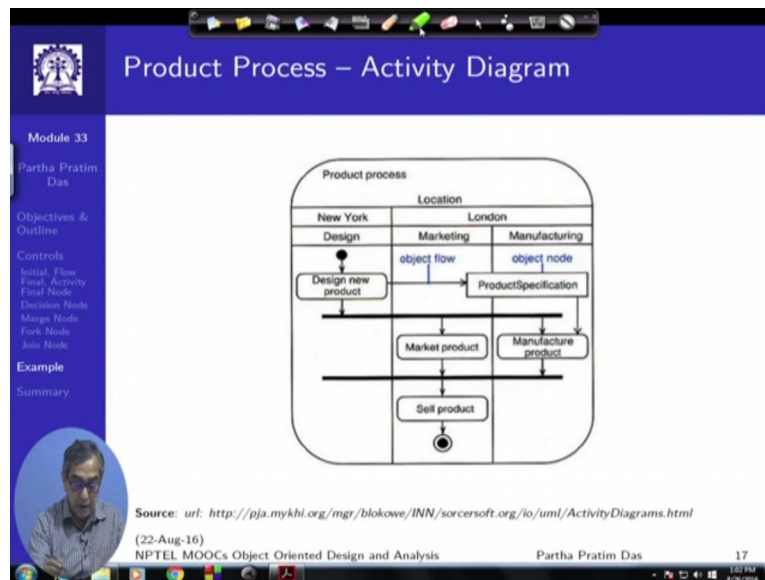
(Refer Slide Time: 19:41)



So, in terms of the online shopping example, we have already discussed the example so I will not spend more time on this. Just we will identify that here this is this certainly is a decision node because 3 different inputs coming and there is only one output. So, this is again is a decision node. this is a am sorry am sorry there are 3 come in and one go out. So actually, this has to be a merge node, this also has to be a merge node because multiple flow come in and one flow is going out.

So multiple flow are getting merged whereas this is a decision node where one flow come in and 2 the waist and found and a non-found this again is a merge node this is a you can see is a decision node and so on. So here we are doing again a merged. So, the through this, these are the only special control nodes that you have inn this diagram, the previous one had that fork and join. So, this is how you can understand the control flow in terms of an activity diagram.

(Refer Slide Time: 20:52)



This is a yet another example of a product processing so here we are trying to show the fork and join kind of control along with the swim lines. So here you can see that there are 3 different swim lines, here and here in terms of design, marketing and manufacturing, the 3 processes of a product and then we have some combining information that both of these happen in London that is a locational information

And this happened in New York though this may not actually be a something that will seriously go into the software system but it is very important to capture this because you know that if this is in London and this is New York any activity that happens across them, any edge that go between them would need to have a kind of external internet based kind of protocol to be followed. So, if we look into the steps of this then we start here first is always to look at the start and the final

And this is designing a new product where it starts and that on this gives an object flow to the product specification. Why it is it an object flow? Because when the design is done you cannot just go to the next activity of manufacturing the product. You have to produce the output of that, so this is an object flow which gives you this object which is the object node. We will see what object nodes are in the next module further but am just showing you the flow.

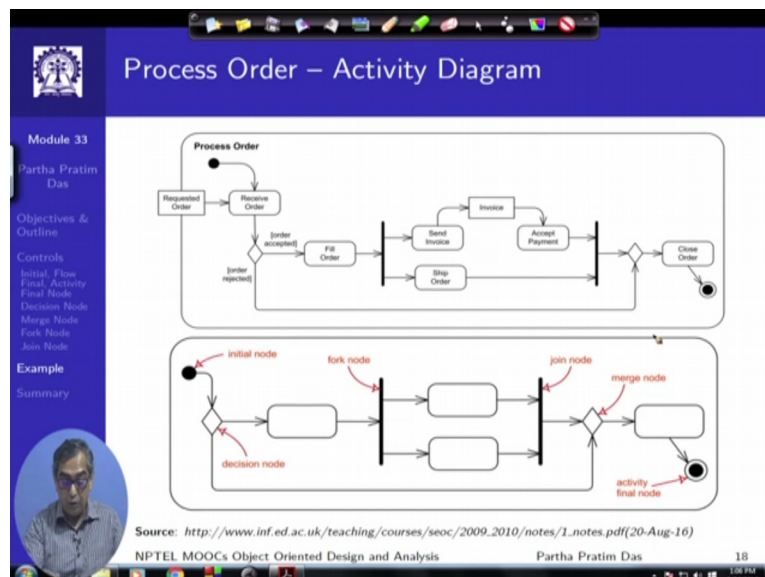
So, after the design I can come to after the design I can come to the manufacture with this object flow edge. Then it goes into a after the design, we go into a fork, this is (E.g.: 22:51) that is the manufacturing of the product and the marketing of the product certainly are done by 2 different teams and must happen together. You cannot market the product say that we have this and manufacture it later you cannot just manufacture the product and keep it and then later on expect that marketing will find out customer.

So, both of these need to happen concurrently, so you have a forking here, so you can see a very interesting structure where this activity will get will need to get initiated from this product specification reaching here and this happening together this control reaching here. So, this designates that the manufacture product activity will need this product specification object when the control comes here and when it initiates in concurrently a market product activity also starts.

Look at so this was the forking and you can we can again look at the merge part of it, certainly when can you go and do the next activity of selling the product. Only when it has been marketing, marketed that the customers know this and only when it has been manufactured. So, both of these edges have to reach this merger, I mean this join point I should write it as a join, it is not merge. So, reach this join point and this will be blocking so if market product is complete this activity is complete.

And product has not been manufactured I cannot go and sell it. In the other way if the manufacturing is complete but it has not completed marketing going to the customer then also I cannot sell it. So, this will be a blocking join which will wait on both these activities to be over and when the token arrives on this as well as this, the token will go to the next cell product and the product will start skating sold. So, this is another kind of activity diagram that we have seen.

(Refer Slide Time: 24:56)



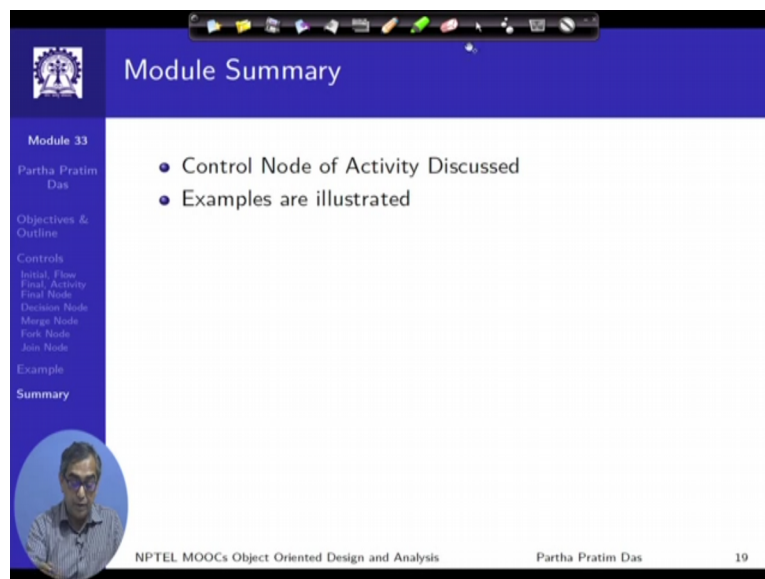
So, finally we are back to the overview diagram that we started with and if we just fill-up fill up some of the details of where it is taken from, this is a processing order. So, you receive the order and this is a decision that is made as to whether the order is accepted or the order is rejected, if the order is rejected you go straight and close the order, that is the end of this activity or if the order is

accepted you fill out the order and then you start 2 different activities in concurrent, sending the invoice and shipping the order.

Sending the invoice is required for getting the payment from the customer, shipping the order process is required to getting the inventory ready. So, sending the invoice will have the next activity as accept payment naturally it involves an object edge flow because the invoice object has to go to the accept payment and only when the payment has been accepted and the order has been shipped, so this is where you again see join is happening only when both of these has happened then the order gets can closed.

So that flows into a merge node where you can so this is a successful order flow and this is a rejected order flow, whenever they merge and then you go to the so you can see very closely as to here by join here anyway merge here and the join needs both of these controls to come in and the merge its either of this control to come in to move to the next activity.

(Refer Slide Time: 26:32)



The image shows a presentation slide titled "Module Summary" for "Module 33" by Partha Pratim Das. The slide is part of an NPTEL MOOC on "Object Oriented Design and Analysis". The left sidebar lists the following topics: Objectives & Outline, Controls, Initial Flow, First Activity, First Node, Decision Node, Merge Node, Fork Node, Join Node, Example, and Summary. The main content area lists two bullet points: "Control Node of Activity Discussed" and "Examples are illustrated". A small circular inset photo of the presenter is in the bottom left corner. The footer includes the NPTEL MOOC title, the presenter's name, and the slide number 19.

Module Summary

Module 33
Partha Pratim Das

- Control Node of Activity Discussed
- Examples are illustrated

NPTEL MOOCs Object Oriented Design and Analysis Partha Pratim Das 19

So, this is these are the different examples, so I would summarize to say that we have discussed particularly different kinds of control nodes in an activity diagram and specifically you should make good understanding of the fork and join the decision fork and join control nodes, because I am sure decision and merge nodes you are more comfortable with because they are the typical flow chart nodes to understand in terms of fork

And join that how activity diagrams can be used not only to represent sequential processes but also to represent concurrent and parallel processes and we have discussed some examples which you can review again and try to build you own solutions.