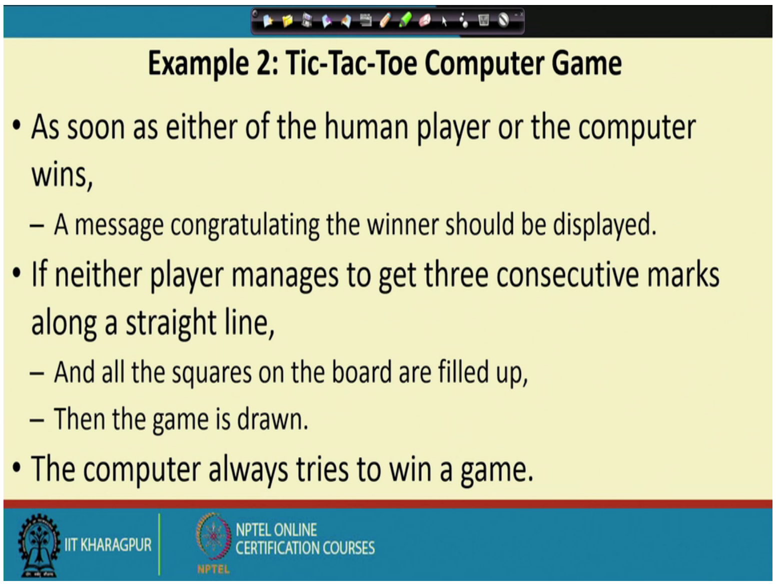
Software Engineering
Prof. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 29
Structure Chart Development

Welcome to this lecture. In the last lecture we had discussed about the transform analysis. Next, we will discuss about transaction analysis and also when to apply transform analysis and when to apply transaction analysis, but before we do that we will just take one more example to see how the transform analysis is applicable? Because, the last example that we saw, that is the root mean square computation is a very simple trivial example, will take lightly more non trivial example and see how transform analysis can be applied.

So, let us look at the Tic-Tac-Toe Computer Game.

(Refer Slide Time: 01:04)



Example 2: Tic-Tac-Toe Computer Game

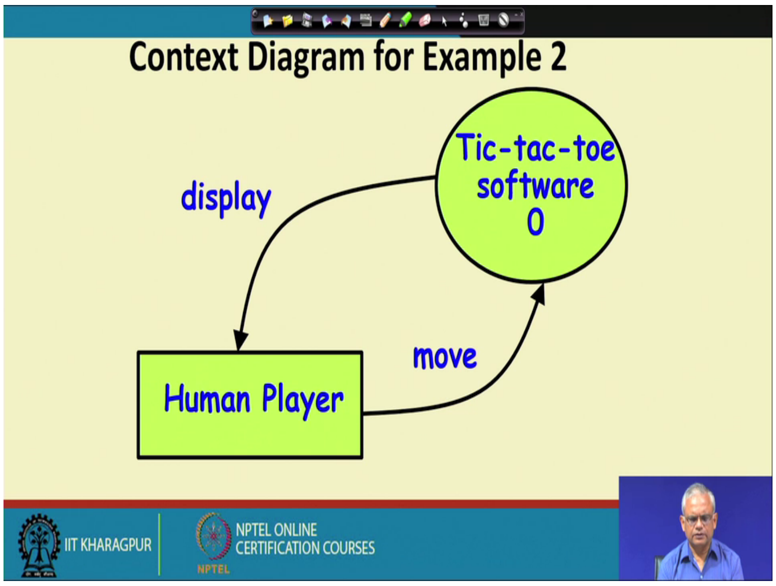
- As soon as either of the human player or the computer wins,
  - A message congratulating the winner should be displayed.
- If neither player manages to get three consecutive marks along a straight line,
  - And all the squares on the board are filled up,
  - Then the game is drawn.
- The computer always tries to win a game.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL

We had developed the DFD representation of the Tic-Tac-Toe Computer Game; we had seen the problem statement earlier, that human player and the computer alternatively make moves. And, once there is a decision, one player wins by marking on conjugative boxes in a row or a column or on the diagonal, winner message should be displayed, but if no one gets that and all the board elements are filled up.

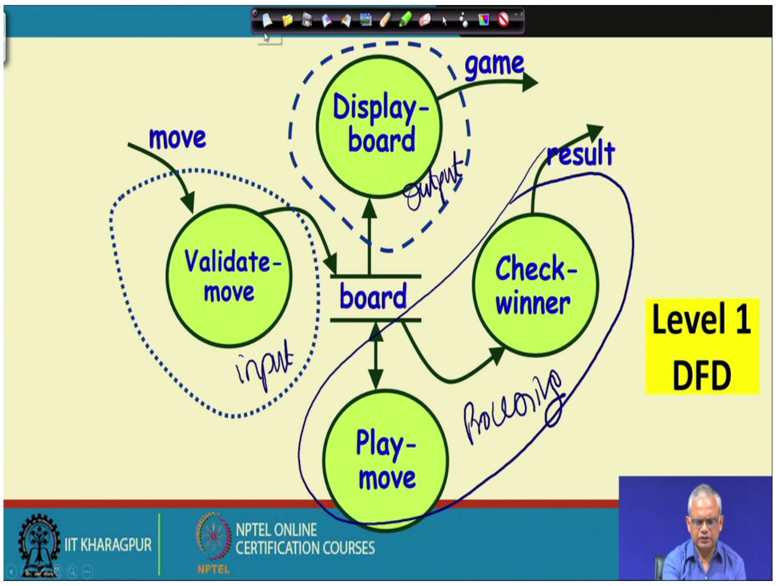
Then the game is drawn and message that the game is drawn is displayed.

(Refer Slide Time: 01:54)


Context Diagram for Example 2
Tic-tac-toe software 0
display
move
Human Player
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

The context level, we had drawn this diagram for the Tic-Tac-Toe software, the human player gives moves to the software. And, it checks whether the human wins then it will display a win message congratulating the human player. Otherwise it makes it is own move checks, if it moves it is move makes it winner, then it will display the corresponding message or it just displays the updated board and prompts the human player to make the move.

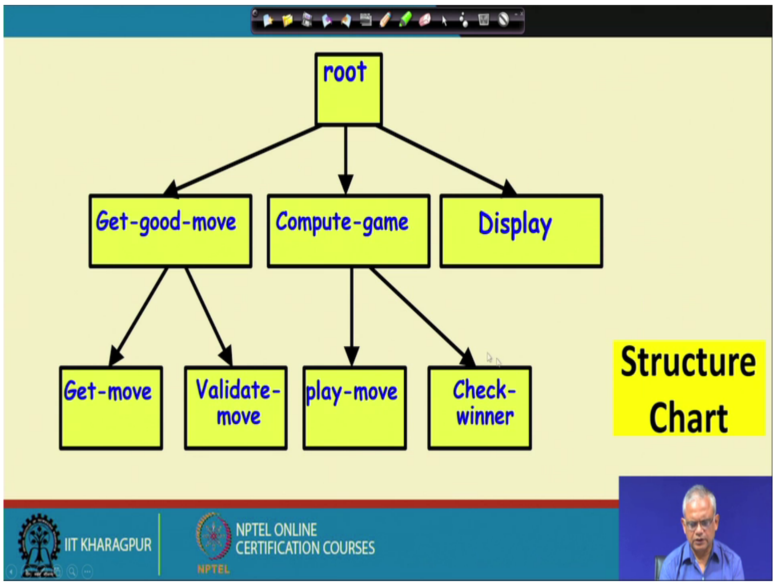
(Refer Slide Time: 02:47)



In the level 1 DFD, we had seen that the move is red and it is validated and then the board data structure is updated. The board data structure is a array of 9 elements, that we had discussed and based on the input the check winner is executed and if there is a result if there is a winner it is displayed otherwise, the computer makes move and finally, the board is displayed. So, how do we do the transform analysis of this? We see that this is the input part, it takes the input and converts physical data into logical form and this is the output part it displays.

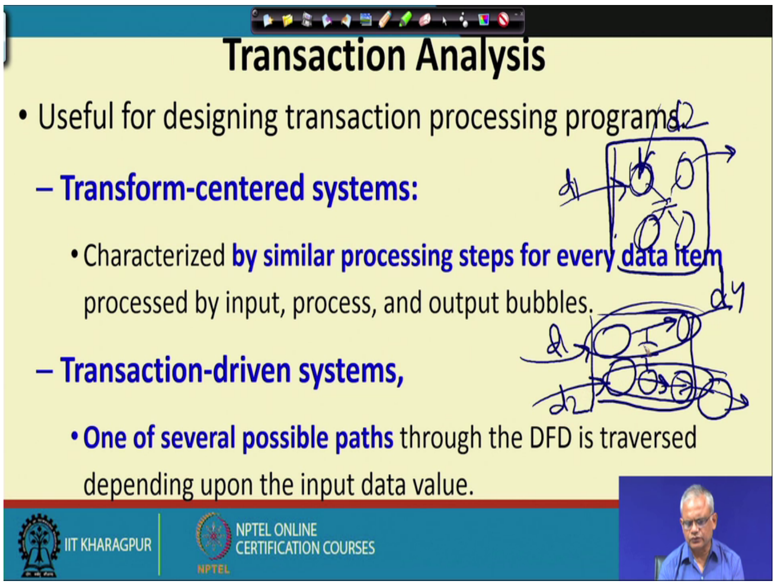
And, the rest are the processing part and even though this bubble here, it produces some output, but then we observe that it is primary role is processing. As, a byproduct of the processing it just displays some result and its task is not to convert logical data into physical form, its main work is to do processing. And, based on that we had classified or we have made this into the processing part. And, once we have done that we can easily draw the module structure.

(Refer Slide Time: 05:03)



So, this is the module structure get good move compute game and display. And, observe here that these are factored because we had get move and validate move are 2 work that is done by their get good move. Similarly, compute game consists of 2 different activities play move and check winner, if you observe the DFD see that these 2 represent, but it is not really necessary that each bubble that is mapped into a module here becomes it is sub module here.

(Refer Slide Time: 05:48)



Transaction Analysis

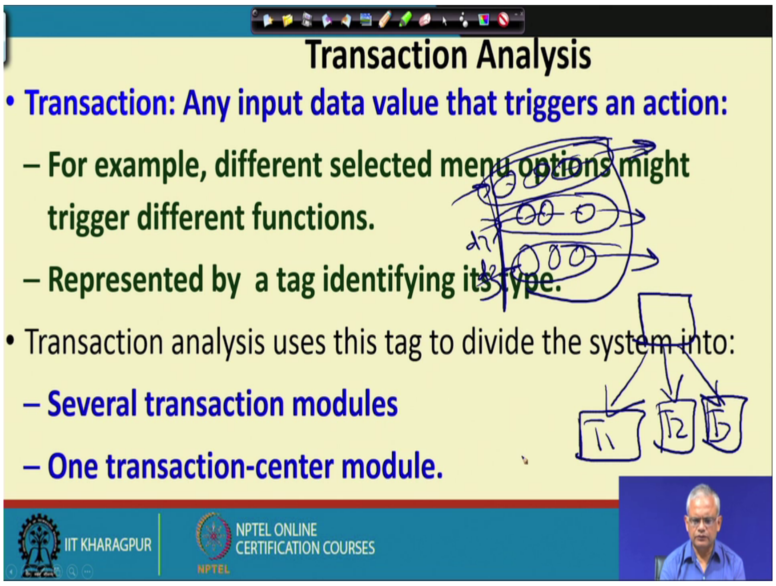
- Useful for designing transaction processing programs.
  - Transform-centered systems:
    - Characterized **by similar processing steps for every data item** processed by input, process, and output bubbles.
  - Transaction-driven systems,
    - **One of several possible paths** through the DFD is traversed depending upon the input data value.

Now, let us look at transaction analysis, but before we look at the transaction analysis.

Let us see a general idea, when to apply transform analysis and where to apply transaction analysis. In transform analysis normally the DFD whatever data is input into the DFD, if this is our DFD. Let us say this is our DFD, whatever data is input they are all use some similar processing and finally, produce the output. The data d 1 if it is input here undergoes similar processing and typically it takes just one data, but it is possible that it may take multiple data items and they undergo similar processing, that is it is processed here and then next processed here and so on.

Where is in a transaction system, some data are processed by this ones. So, this is one transaction and another data may be processed through a different transaction sorry different bubble. So, each of these will become a transaction. So, d 1 data is processed through these bubbles and d 2 through these bubbles. So, they are 2 different transactions.

(Refer Slide Time: 07:50)



Transaction Analysis

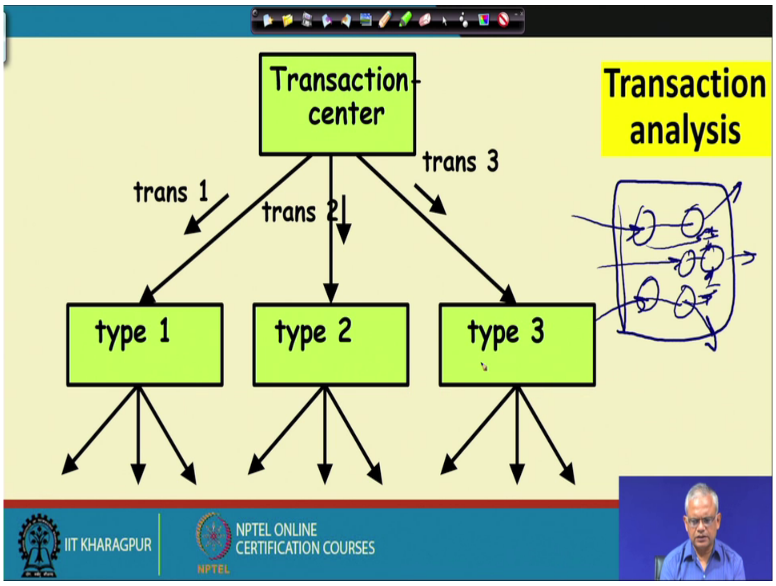
- Transaction: Any input data value that triggers an action:
  - For example, different selected menu options might trigger different functions.
  - Represented by a tag identifying its type.
- Transaction analysis uses this tag to divide the system into:
  - Several transaction modules
  - One transaction-center module.

Typically, when there are menu options depending on the menu option chosen some functions are executed.

For example, if you want to let us say a drawing package, you want to do editing some specific functions will be invoked. If you want to do printing some other functions will be drawn they will be executed, if you want to do save some other functions will be executed. So, typically the data that is input by to a DFD, they are processed in by say different functions and each of those functions we call as a transaction and we can represent this in the form of a structure chart and typically, if we identify in a DFD.

That these are one transaction and this is another transaction. So, this data is processed this functions, this data is processed this functions and this data processed this functions. Then, there are 3 transactions here we do a in the structure chart representation, we have a root module or the transaction centre module and this 3 become the transactions here; T 1 T 2 T 3. And, then we again do a transaction analysis or a transform analysis on each of the transactions. And finally, we will get a module structure, will see the details as we proceed.

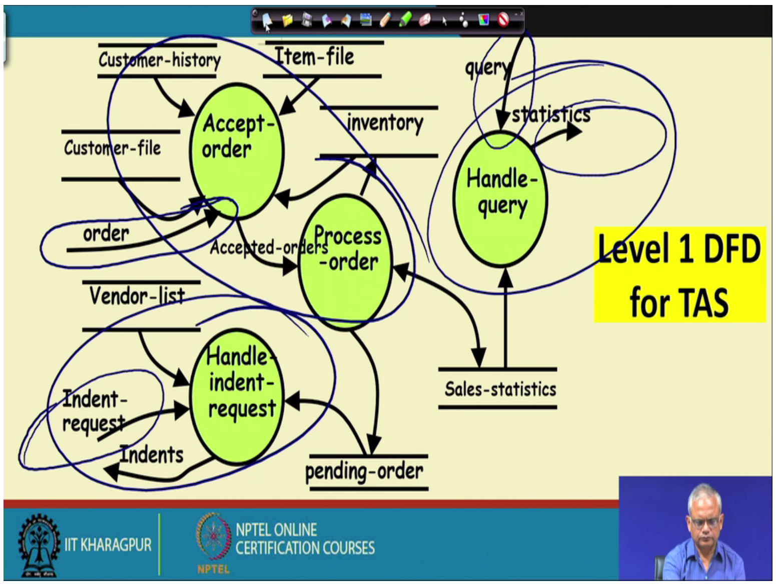
(Refer Slide Time: 10:07)



So, the main point here is that if we observe a DFD and see that it takes different data, we observe a DFD and find that different processes here, take data and these data are processed through different processes. Then, we call this as the transactions and this process maybe takes data here produces output may be they share some data here, it produces data and this is shared and so on.

So, we identify the transactions here and we write a transaction centre module and then we draw the 3 transactions, and this each transaction needs to be decomposed into further modules.

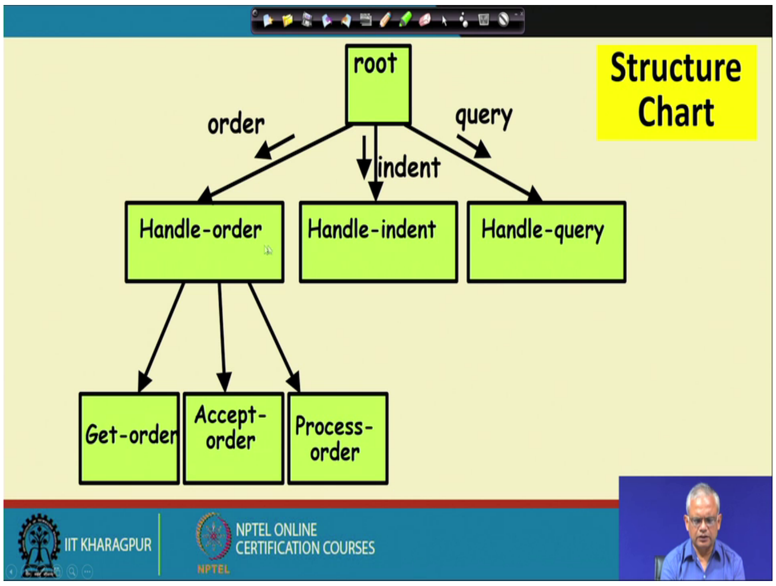
(Refer Slide Time: 11:26)



For example, if we have this kind of a DFD. We see here that it takes many data items. For example, it takes one data item here indent request and it is a processed here based on the pending order and it produces an output. Whereas a query is a data item which is processed through this and the output is produced and the order is another item, data item which is processed through these 2 and an output is produced.

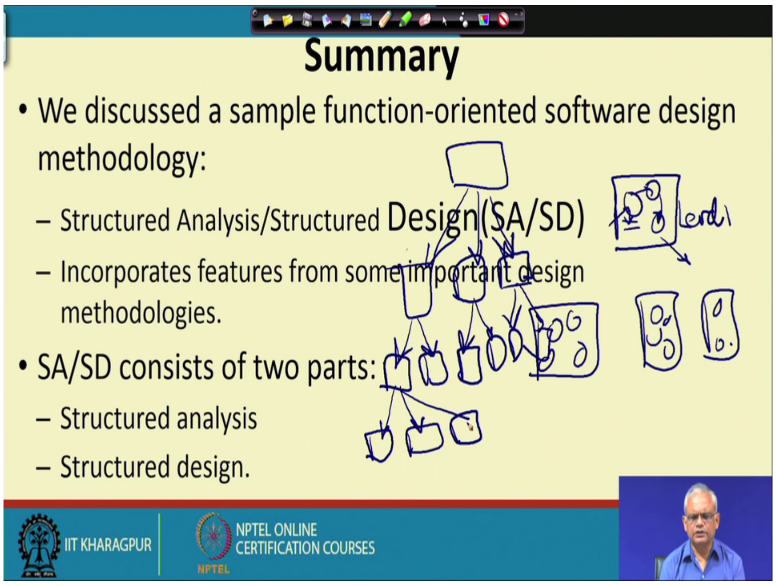
So, we will identify 3 transactions here and once we identify the transaction we will write transaction centre module and then the next level will draw the 3 transactions.

(Refer Slide Time: 12:28)



So, this is what the 3 transactions root, handle order, handle indent, handle query and then we factor them get order, accept order and process order.

(Refer Slide Time: 12:43)



Summary

- We discussed a sample function-oriented software design methodology:
  - Structured Analysis/Structured Design(SA/SD)
  - Incorporates features from some important design methodologies.
- SA/SD consists of two parts:
  - Structured analysis
  - Structured design.

So, we saw that the structure chart is obtained as a result of the structured design, we first do a structured analysis given a problem, during the procedural design step we will do a structured analysis, where we look at the requirement specification and then we develop the functional decomposition.

We come up with the DFD model, typically consists of many levels and then the next step we do the structured design. In the structured design we take the DFD representation and for each DFD we identify whether there is a; we apply transform analysis or transaction analysis. Typically, very simple processing that is the lowest level of the DFDs they become transform analysis. So, if we have a DFD which is the level 1 DFD and then we based on the level 1 DFD, for each of the bubble here in the level 1 DFD, we represent we develop the level 2 DFD and then the level 3 DFD if necessary and so on.

So, given a design like this the context diagram I do not really use, while applying the structured design start with the level 1 diagram, and unless it is a very simple system like let us say the Tic-Tac-Toe or this computer RMS etcetera very trivial software. Typically the level 1 DFD we observed that transform analysis needs to be performed.

Only for very simple software level 1 DFD will lead to a transform analysis, but typically transaction analysis needs to be done for any non trivial software. Because a non trivial software will provide different functionalities just observe that both the compute RMS and the Tic-Tac-Toe, the user there were only one user and could give only one type of move.

But, when there are multiple users like let say library software or any non trivial software. We will see that there are multiple users and also each user can give different types of data to the system. For example, a user in a library software might issue a book or return a book or search for a book and based on which functionality in books, different bubbles will get activated.

And therefore, the level 1 DFD for any non-trivial software will normally be a transaction processing. So, we will have to apply transaction processing and even the level 2 is often a non-trivial software a transaction processing different transactions were possible here.

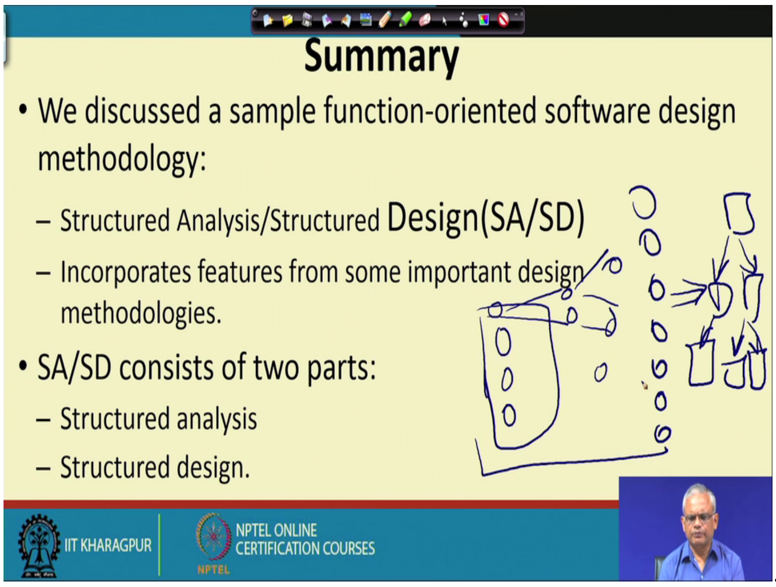
And, normally at the lowest level of the DFD, the transform analysis is applied, because those represent the simplest processing. And, once the module structure is obtained, because through all these DFDs will enhance the module structure will identify the root here and the transactions here. And, then the processes that are identified here for transaction, we look at the corresponding DFDs and do the next level of the structure chart and so on.

And, then we look at the next level. So, this we do the structure chart the transform analysis were corresponding to the bubbles which map on to this and so on. And, until we reach the lowest level DFD and there we will see that invariably the lowest level DFD is a transform analysis.

Where we have this typical read input, do some processing and produce output. We looked at the basics of procedural design, where starting with the requirement specification. We, saw by taking the requirement specification we identify the functions and draw the DFD model. And, from the DFD model how to get the structure chart or high level design? And, once we get the high level design we do the detailed design by for each module we develop the module specification in terms of the data structure and the algorithms.

And the next topic, we will take up is the object oriented design. The object oriented design is completely different then the procedural design, because we will see that the procedural design is a top down design. It start with this function specified in the requirements documents split them, decompose them into soft functions and so on. And finally, we get the structure chart representation.

(Refer Slide Time: 20:06)



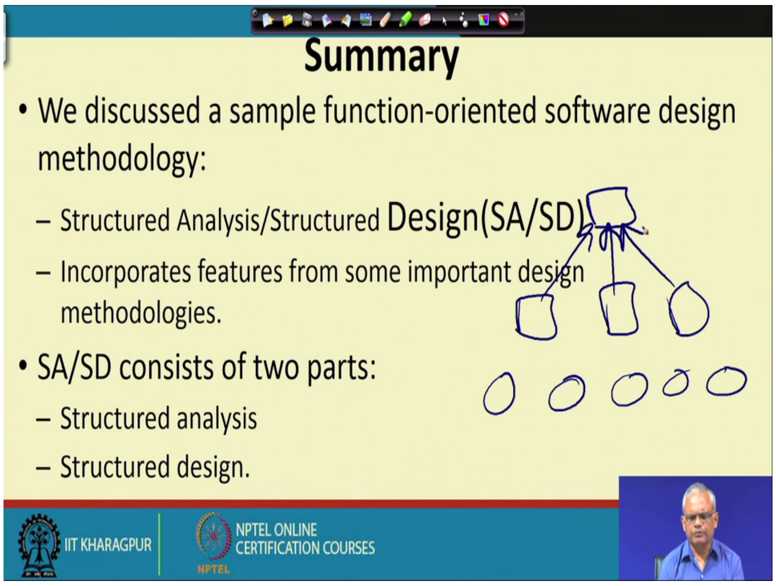
Summary

- We discussed a sample function-oriented software design methodology:
  - Structured Analysis/Structured Design(SA/SD)
  - Incorporates features from some important design methodologies.
- SA/SD consists of two parts:
  - Structured analysis
  - Structured design.

So, we start with the functions represented in the requirements document and then achieve the decomposition of that and at the end of the data flow diagram we get a set of simple functions and through a structured design, we map this into a module structure.

So, the procedural design is a top down design on the other hand will see that in the object oriented design, we have a bottom up design.

(Refer Slide Time: 20:55)



Summary

- We discussed a sample function-oriented software design methodology:
  - Structured Analysis/Structured Design(SA/SD)
  - Incorporates features from some important design methodologies.
- SA/SD consists of two parts:
  - Structured analysis
  - Structured design.

In the object oriented design, we will first identify the objects and then, once we identify these are the objects we will build the classes out of this. And, then will build the class relations between this in terms of aggregation, in terms of inheritance and so on. And, then will do few other diagrams like structure chart and so on and then we will get the code structure.

So, we can say that the procedural design is a top down design where as the object oriented design is a bottom up design. So, the 2 design techniques are appear to be entirely different; one is a procedural top down approach, the other is a bottom up approach, but then we will see that even during the object oriented design. We, will see that we will use the concepts of the procedural design, because after all for every class will have methods. And, those methods some of the methods are very complex method and we need to design those using the ideas from procedural design.