

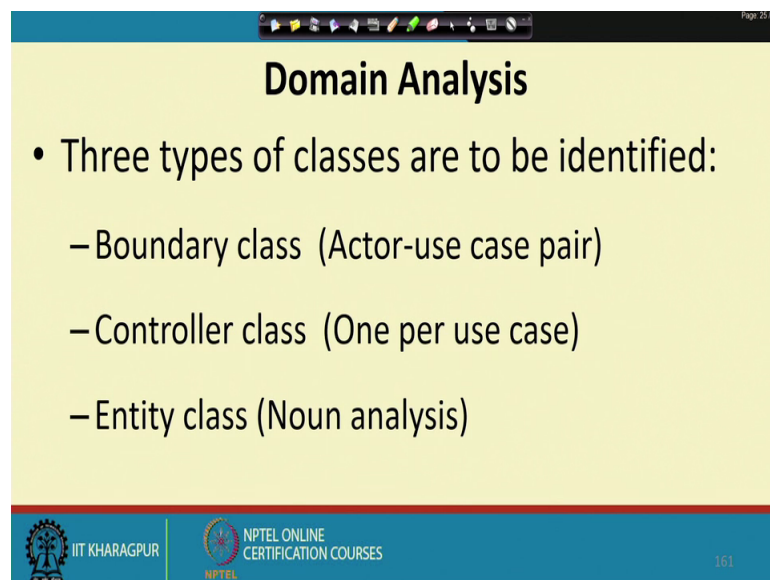
**Software Engineering**  
**Prof. Rajib Mall**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture – 41**  
**Domain Analysis**

Welcome to this lecture. In the last lecture, we were discussing topics in object oriented software design. And, we had seen that given a problem description the first thing we need to do is the use case diagram. The use case model is the central model and it portrays the customer's perspective of a problem.



And, once the use case diagram has been successfully completed, that sets the ground floor doing the Domain Analysis. In the domain analysis we try to develop the first cut class diagram. And, we had were discussing, how to go about doing the domain analysis? And, we said that from the problem description and the use case model we can identify 3 types of objects. From the use case model, we can identify the boundary objects and the controller objects. And, then from the problem description we need to identify the entity objects. So, now, with that background let us proceed in this lecture.

(Refer Slide Time: 01:35)



**Domain Analysis**

- Three types of classes are to be identified:
  - Boundary class (Actor-use case pair)
  - Controller class (One per use case)
  - Entity class (Noun analysis)

 IIT KHARAGPUR |  NPTEL ONLINE CERTIFICATION COURSES

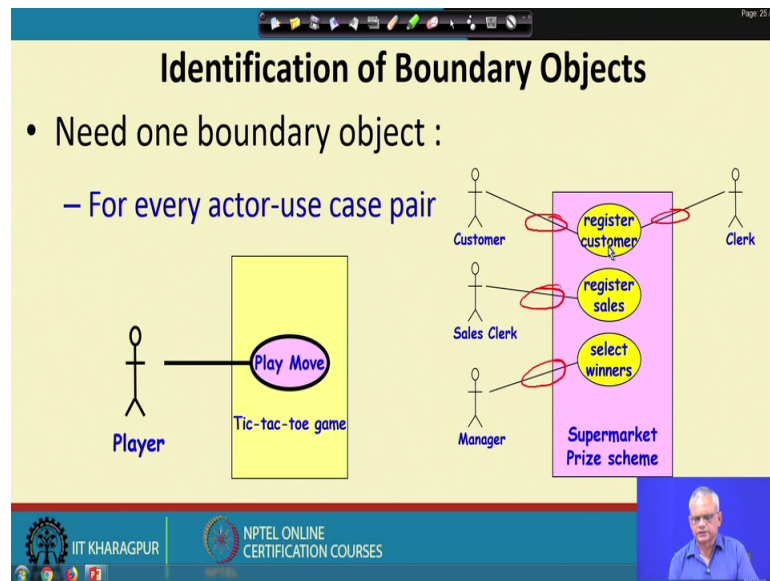
161

The domain analysis aim is to develop the first cut class diagram and in the subsequent steps in the object oriented design, we will refine this diagram. And, in domain analysis we are interested to identify 3 types of objects the boundary class. And, the boundary

class we identify from the use case diagram and every actor use case pair becomes a boundary class.

The controller class identification is also straightforward, we identify one per use case, you look at the use case diagram identify how many use cases are there? And, we need that many controller classes. And finally, the entity classes which we identify by doing an noun analysis on the problem description.

(Refer Slide Time: 02:41)



So, as you can see the boundary class and the controller class are rather straightforward to identify. Just from an inspection of the use case diagram whereas, the entity classes require bit more experience and practice to be able to successfully identify the entity classes.

Now, let us try to identify the boundary classes for the use case diagrams, which you had already developed. Let us look at the Tic-tac-toe game use case and this is the simple problem and we have only one use case play move. And therefore, we have just one boundary here. And therefore, we need only one boundary class. On the other hand for the supermarket prize scheme, we have 3 use cases and we have 4 actors.

And therefore, there are 4 actor use case pair, one boundary between the register customer and customer another boundary between the register customer and clerk, a boundary between register sales and sales clerk and boundary between the manager and

the select winner. So, we need 4 boundary classes. We can name these boundary classes as customer register customer boundary, sales clerk register sales boundary, manager select winners boundary and register customer clerk boundary.

And, there are 3 controller classes, register customer controller, register sales controller, and select winner controller. And, as we had discussed in the last lecture the boundary is for the user interface, for the user input and output to the user taken care by the boundary class. Whereas, the controller class has the overall control of executing the use case once the user initiates the use case the controller class has business logic. And it coordinates several other classes including entity classes to finally, provide the result for the use case.

(Refer Slide Time: 05:39)

The slide, titled "Identification of Controller Objects", contains the following content:

- Examine the use case diagram:
  - Add one controller class for each use case.
  - Some controllers may need to be split into two or more controller classes if they get assigned too much responsibility.

The diagrams illustrate these concepts:

- A use case diagram for "Tic-tac-toe game" with a "Player" actor and a "Play Move" use case.
- A use case diagram for "Supermarket Prize scheme" with three actors: "Customer", "Sales Clerk", and "Manager". The use case is split into three sub-use cases: "register customer", "register sales", and "select winners".

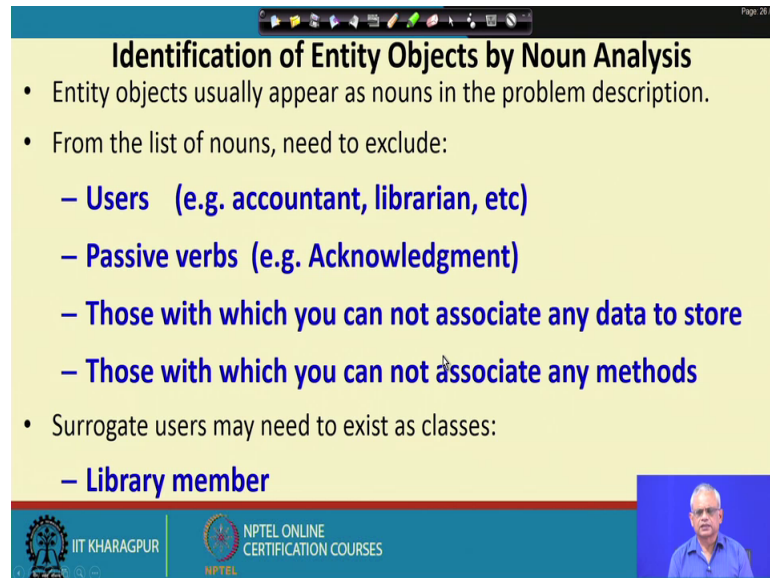
The slide footer includes logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, along with a small video inset of a speaker.

The controller classes are also easily identified from the use case diagram. For every use case we need a controller and the controller has the business logic and is an overall control of realising the behaviour of the use case by coordinating the actions of several other classes. But, then some of the controllers may become extremely complex. They might have to coordinate the actions of dozens of other objects.

And in that case to simplify the design, later in the design steps we might split them into two or more controller classes, but then you might ask the question that how do we know that a controller class has become extremely complex? The answer to that question is that as we go to the next step of the design that is developing the sequence diagram we will find that for those classes, which are very complex behaviour?

The sequence diagram becomes extremely complex. There large number of message exchanges with many objects. And, becomes easy by just inspecting the sequence diagram for use case to see the controller class become very complex we will examine that in the next step. And, similarly for this example there are 3 controller classes that are required.

(Refer Slide Time: 07:28)



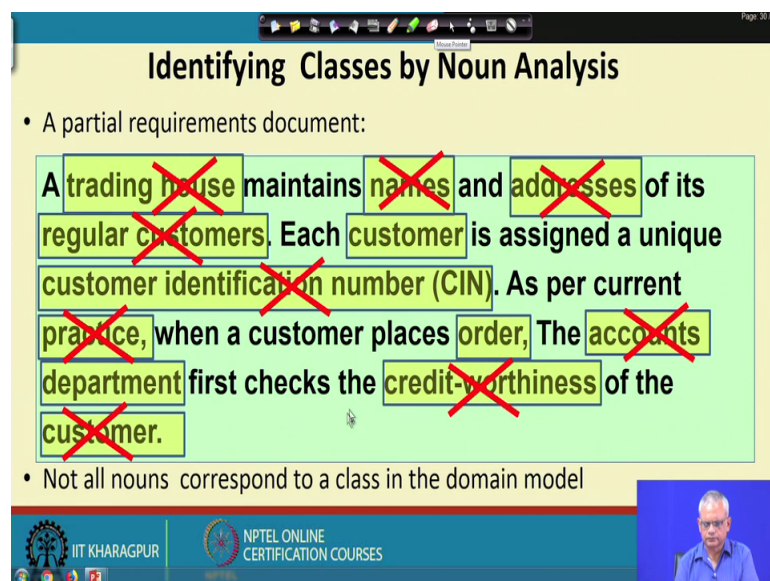
**Identification of Entity Objects by Noun Analysis**

- Entity objects usually appear as nouns in the problem description.
- From the list of nouns, need to exclude:
  - **Users** (e.g. accountant, librarian, etc)
  - **Passive verbs** (e.g. Acknowledgment)
  - **Those with which you can not associate any data to store**
  - **Those with which you can not associate any methods**
- Surrogate users may need to exist as classes:
  - **Library member**

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, let see how to go about identifying the entity object by noun analysis?

(Refer Slide Time: 07:37)



**Identifying Classes by Noun Analysis**

- A partial requirements document:

A trading house maintains names and addresses of its regular customers. Each customer is assigned a unique customer identification number (CIN). As per current practice, when a customer places order, The accounts department first checks the credit-worthiness of the customer.
- Not all nouns correspond to a class in the domain model

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

To perform the noun analysis, we examine the problem description and then identify the nouns, the entity objects are usually nouns in the problem description. But, then if we identify the nouns in the problem description not all nouns correspond to entity classes. We need to exclude several of the classes. For example, we might have to exclude the users. For example, if there is a noun let say accountant or librarian, we need to exclude we need to exclude passive verb forms, the passive verb forms appear like noun like acknowledgment.

But, then we know that these are not really nouns. And, we exclude the and also from the list of nouns that we identify we check, if we can associate some data for it to store. And, we can associate some methods with it, if we cannot associate any data or methods with it, then this cannot be classes because every class after all must have some data and methods.

This is the general guideline about once we identify the nouns, we need to exclude some of the nouns during the noun analysis, but one thing we must mention that some of the users can be actually classes entity classes. This occurs when we have to remember some aspects of the users. For example, let say library member in a library automation system needs to be a class, because we need to remember how many books are library member has taken? When does he return are their fines outstanding? So, sometimes the users may become classes and we call them as surrogate classes for users.

(Refer Slide Time: 10:28)

• Remember that a class represents a group (classification) of objects with the same behavior.

- **We should therefore look for existence of similar objects during noun analysis**

• Even then, class names should be singular nouns:

- Examples: **Book, Student, Member**

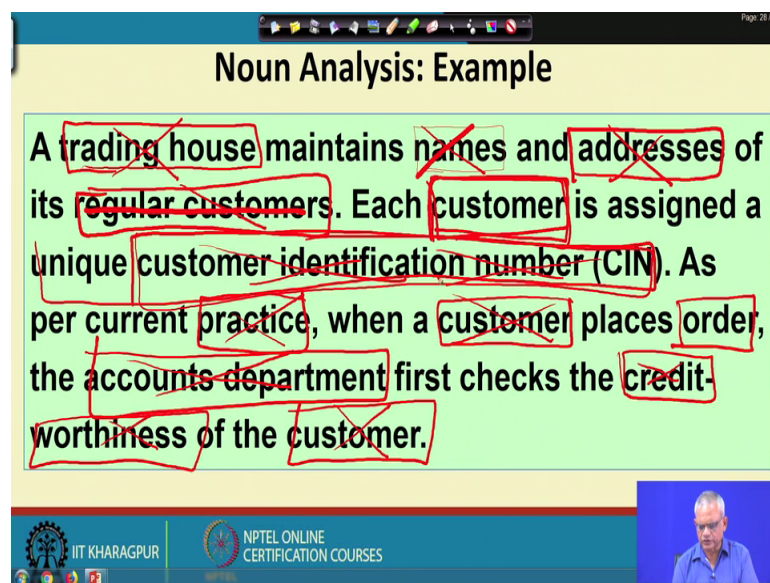
**Identifying Classes**

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Another hint is that entity classes normally appear as groups, there is several objects which are very similar. For example, let say book in a library a thousands of books. So, we have thousands of objects which are similar and let say member thousands or hundreds of members and all are similar. And, that is a characteristic of entity classes that normally they occur in groups a large number of them, but even though we know that there are many class corresponds to many objects.

But, the we normally name a class as a singular. For example, we name even though it represents the objects are hundreds or thousands of objects we name the class as book. Similarly there may be tens of thousands of students, but then the name of the class is student name of the member class is member. So, the class name is normally singular even though we create many instances from the classes.

(Refer Slide Time: 12:10)



The slide is titled "Noun Analysis: Example" and contains the following text: "A trading house maintains names and addresses of its regular customers. Each customer is assigned a unique customer identification number (CIN). As per current practice, when a customer places order, the accounts department first checks the credit-worthiness of the customer." The nouns in this text are circled in red: trading house, names, addresses, regular customers, customer, unique customer identification number (CIN), customer, places order, accounts department, credit-worthiness, and customer.

Now, let us take an example to see, how the noun analysis is performed? But, let me just tell that even though here we really do the noun analysis, but as you gain some experience of solving a few dozens of problem or at least say a dozen of problem. You do not have to really do the noun analysis identify all nouns and then eliminate the ones which are not classes, but with experience you can read through the text and mentally identify, which are entity classes, but for the first few problem description.

We actually need to do the noun analysis and then you can get the hang of it that is now which type of noun we are looking for. And you can mentally note down those at you

read of a problem description, but now for this simple example let us do noun analysis. And, this is the example for which we had done the use case diagram. Now, let us do the noun analysis for this part of the problem description the first is to identify all nouns.

So, we identify trading house is a noun to identify names, addresses, regular customer, customer, unique customer identification number ok, let me just do customer identification number. Unique is a qualifier to this practice, customer, order, accounts department, credit worthiness and customer. So, these are the nouns in this text and we know that many of these nouns are not really corresponding to entity classes. And, we need to eliminate those and only with those entity classes, which represent a group of objects and you can associate some data and method those are the entity classes.

Now, trading house is not really an entity class, because it is the name of the problem. Names and addresses are actually attribute we cannot associate any further methods or data with names and addresses. So, these are attributes rather than classes. Regular customer and customer they are actually synonyms, we just return here customer they are actually the regular customers. So, we will just delete one of that and we will just return the customer. This is a possible entity class because there are many occurrences of a customer. And each customer we need to store the details. For example, what is the name address credit worthiness and so on?

Customer identification number is just a part is attribute of the customer rather than being a class. So, I will cross it, practice is a noun form of a verb I will just cross it. Customer we have already identified cross it order yes there are many orders. And, we can associate some data and methods with order and this is a possibility. Department accounts department is not the name of a department and it is actually user, credit worthiness is an attribute of the customer. So, after eliminating all that we see that we are left with only the customer and this is again the same customer.

So, we have the customer and order these are the two only that I have been identified. This is the same problem description just repeating, what I said first need to do the noun analysis need to identify all the nouns? So, trading house names, addresses, regular customer, customer, customer identification number, practice, customer may be I just left that maybe we should have the customer identified again here, but then of course,

already identified customer once. So, this is just a duplicate accounts department credit worthiness and again customer.

Now, we need to ignore or delete those, which are either users or with which you cannot assign any data or methods. So, trading house can eliminate this is just the name of the software. And these are the attributes were we cannot assign any further methods data with this can eliminate, that regular customer and customer are synonyms we just need to keep one of that delete one. Customer identification number is an attribute is not a class; we cannot assign any methods or further data with customer identification number.

And therefore, delete it practice is a noun form of a verb delete it, order yes accounts department we delete it, it is a user who checks the credit worthiness and credit worthiness is an attribute. And, customer already selected and therefore, we delete it here also. So, finally, you are left with customer and order.

(Refer Slide Time: 19:49)

The slide is titled "Identification of Entity Objects" and contains the following content:

- Usually:
  - Appear as data stores in DFD
  - Occur as group of objects that are aggregated
  - The aggregator corresponds to registers in physical world

The diagram shows two aggregation relationships. On the left, a box labeled "SalesHistory" is connected to a box labeled "SalesRecords" by a line with a diamond at the "SalesHistory" end and a "1" near the diamond, and an "\*" near the "SalesRecords" end. On the right, a box labeled "CustomerRegister" is connected to a box labeled "CustomerRecord" by a line with a diamond at the "CustomerRegister" end and a "1" near the diamond, and an "\*" near the "CustomerRecord" end.

The slide footer includes the IIT KHARAGPUR logo, the NPTEL ONLINE CERTIFICATION COURSES logo, and a small video inset of a man speaking.

Normally, the entity objects they occur in a group many of them. For example, let say sales record or customer record there are many objects that are get created. And, we normally have another class which aggregates all this objects and it in fact, creates all this objects.

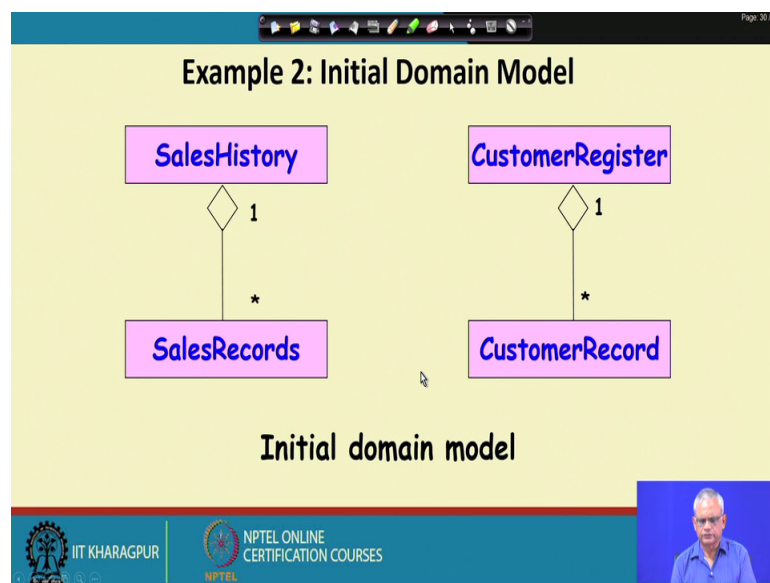
So, for sales records we have an aggregator here sales history. Similarly for the customer record there are many customer records and the aggregator is the customer register. The



customer register keeps track of customer record it in fact, creates the customer record. Almost every entity class have this form that is aggregated and multiple objects created for the sales record.

If, we can look at the DFD representation of the problem, then the entity objects appear as D data stores in a DFD. These are group of objects that are aggregated and the aggregator typically corresponds to a register in the physical world, which keeps track of this large number of objects that are created.

(Refer Slide Time: 21:27)



So, once we will create the entity classes, we call it the initial domain model. And, then we add the boundary and the controller classes and then we go for the next step that is the sequence diagram development from the domain model

(Refer Slide Time: 21:48)

**Example 1: Tic-Tac-Toe Computer Game**

- A human player and the computer make alternate moves on a **3X3** square. *Board*
- A ~~move~~ consists of marking a previously unmarked square.
- The user inputs a number between 1 and 9 to mark a square
- Whoever is first to place three consecutive marks along a straight line (i.e., along a row, column, or diagonal) on the square wins.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, let us do few practice few. And then see whether we can do the domain model the Tic-Tac-Toe Computer Game already done the use case diagram, but just to refresh your memory just quickly read it. A human player and the computer make alternate moves on a 3 by 3 square move consists of marking a previously unmarked square, the user inputs a number between 1 and 9 to mark a square, and whoever is first to place three consecutive marks along a straight line that which may be along a row column the diagonal wins.

(Refer Slide Time: 22:35)

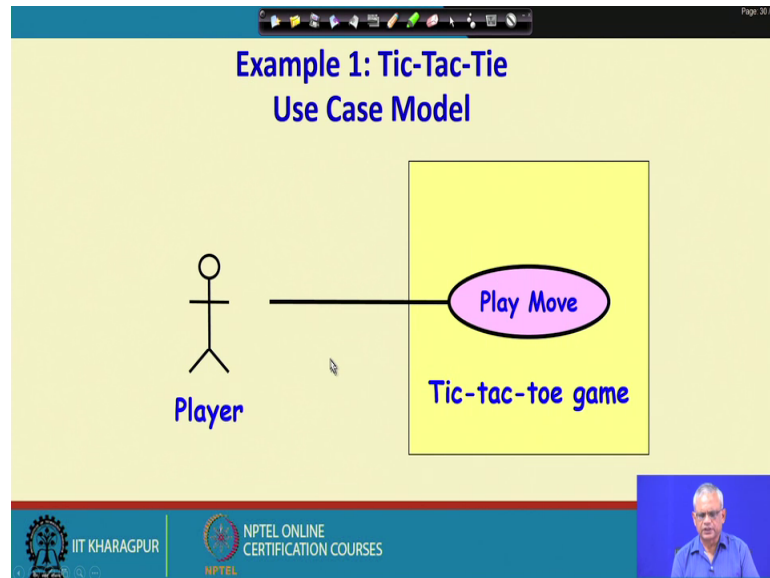
**Example 1: Tic-Tac-Toe Computer Game cont...**

- As soon as either of the human player or the computer wins,
  - A message announcing 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

And, whenever human player or computer wins message announcing the winner is displayed, but it may so, happen that all the squares are adjusted and no player manages to get three consecutive marks along a straight line. In this case the game is drawn; the computer always tries to win a game. Now, let us see how to go about developing the domain model we have drawn the use case diagram for this problem.

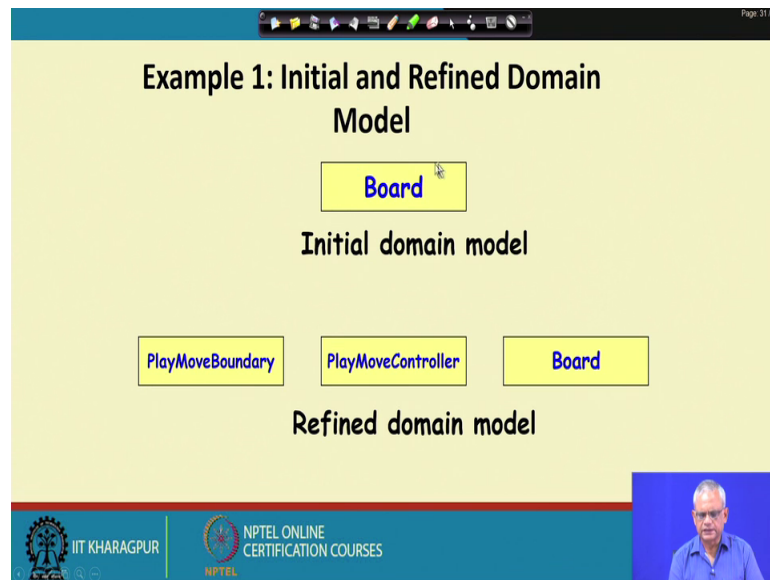
(Refer Slide Time: 23:13)



The use case model appear like this, that is only one use case play move, we write the name of the software the use case and then the actor is the play. Now, the next step is to develop the domain model. For developing the domain model identifying the controller class and the boundary class in straightforward, we just have one boundary and one controller. But, the major problem that we need to address is how to identify the entity classes. Let us revisit the problem again; we need to do the noun analysis of this. And, we can see here we can eliminate mentally human player computer move and then you can see the 3 by 3 square possible move eliminate move also.

Marking previously unmarked square, unmarked is the attribute of a square. The user inputs number between 1 and 9 to mark a square and whoever is the first to place three consecutive marks along a straight line square wins; So, based on this we can identify this 3 by 3 square to be the entity class and we will call it as the board. So, the board consists of 3 by 3 square and each square has the attribute whether it is marked or unmarked.

(Refer Slide Time: 25:41)



And, once we identify this becomes straightforward to draw the entity diagram only one entity class that is the board. And, then we refine it we add the boundary and the controller classes. So, finally, we have 3 classes that have been identify as the part of the domain model and in the next step we go for the sequence diagram from the domain model. And by drawing the sequence diagram, we will be able to not only identify the methods each class will have we will also identify any class relations between them. And also we will identify the attributes of each of these classes.

We will take one more example just to have more practice on identifying the domain model, but now we almost at the end of the lecture we will stop now and continue in the next lecture.

Thank you.