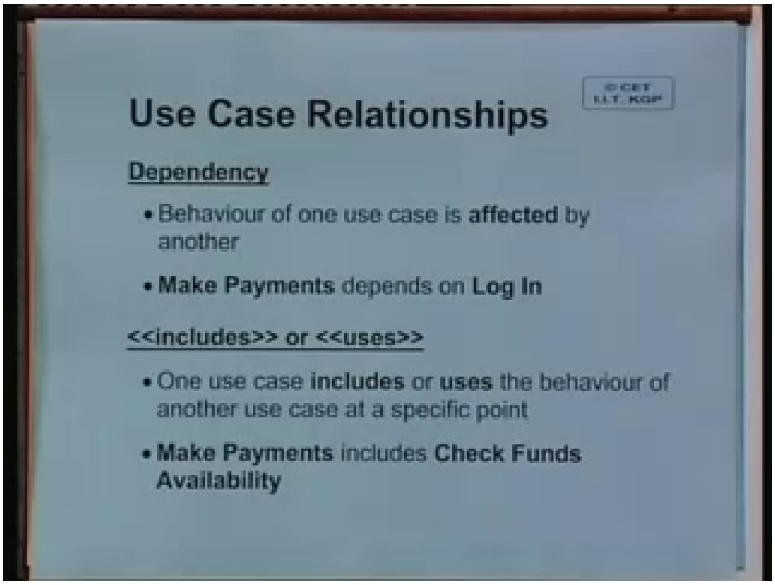
Management Information System
Prof B. Mahanty
Department of Industrial Engineering and Management
Indian Institute of Technology, Kharagpur

Lecture No. #29
OOAD – III

We were discussing about object oriented analysis and design particularly about the use case diagrams and use case analysis. So, we have identified that there are various forms basically, there is something called a use case diagram, but, the diagram not everything. Apart from the diagram only a very broad kind of an over view of the thing and however, a much important thing about use cases at the use case descriptions. The sequence of events that occurs and the actor relationships and so and so forth has also, to be given in great detail. But there are certain things that the diagram and also depict.

(Refer Slide Time: 01:49)

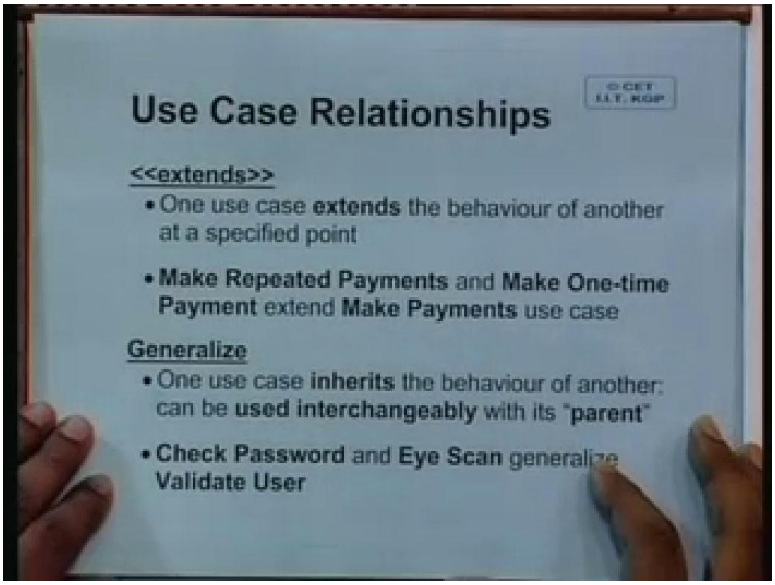


So, let us try to see what are those use case relationships which can actually, be possible when it comes to the use case diagram. You can show something like a dependency, dependency that is a behavior of one use case is affected by another. So, basically when there are two use cases. You see we are not talking about actors the actors are involved in an use case say for example, registration requires a student as well as the registration the courses you know which are a catalogue of course,s. So, course catalogue one can say a. So, in this case the course catalogue and the student could be two actors.

Now, say for example, log in log in as an example and the make payments. So, you cannot see we were discussing about a case where you are having an account in a bank and from this bank you are making online payments are possible but, the your customers all have an account and the customers are making their payments through an online transaction. But now, when the customers are logging in you see until you log in you cannot do make payments right the payments cannot be made. So, they are a is a kind of a dependency much more important are the use of includes or uses. What happens? One use case includes or uses the behavior of another use case at a specific point.

So, make payments includes check funds availability basically, see make payments is an use case, but, making payment is a bigger use case. As I said that use case usually when you are defining use cases try to see some sort of chunks of activity not just a single activity. But sometimes what happens to make it a even more understandable you may have to go to the detailed a description of the thing. So, like that you have the make payments, within that you may have the check fund availability. So, these are called the includes or uses we will we will explain these after some time.

(Refer Slide Time: 04:20)



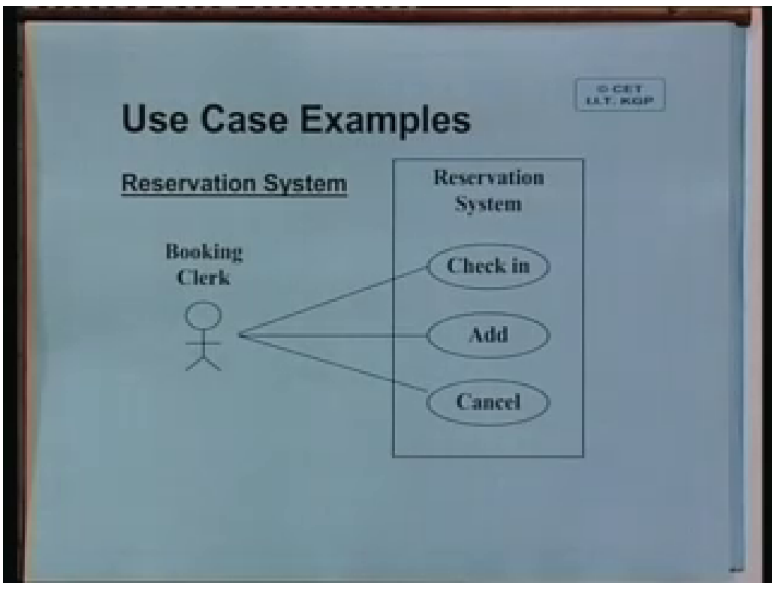
See the extends is another very important thing one use case extends the behavior of another at a specified point for example, make repeated payments and make one-time payment extend make payments use case. So, basically there are two types of payments. One is a repeated payment when you try to pay let us say a over a different installments

number of installments that you may call a make repeated payments and make one-time payment. So, one-time payment and the repeated payment both are basically kind of extension to the make payments use case. Finally, we have the generalize the basic idea of generalize is one use case inherits the behavior of another and therefore, can be used interchangeably with its parent.

So, for example, when you are a validating user the validating user can actually be possible either through check password or through eye scan. So, when the user is physically available you can always a scan him with your eyes and you can see whether, it is a correct user or you can check password. So, this is usually called the generalization. So, the generalization is you can see that keep coming back in many different ways because generalization is a one of the very important constructs of object oriented concepts.

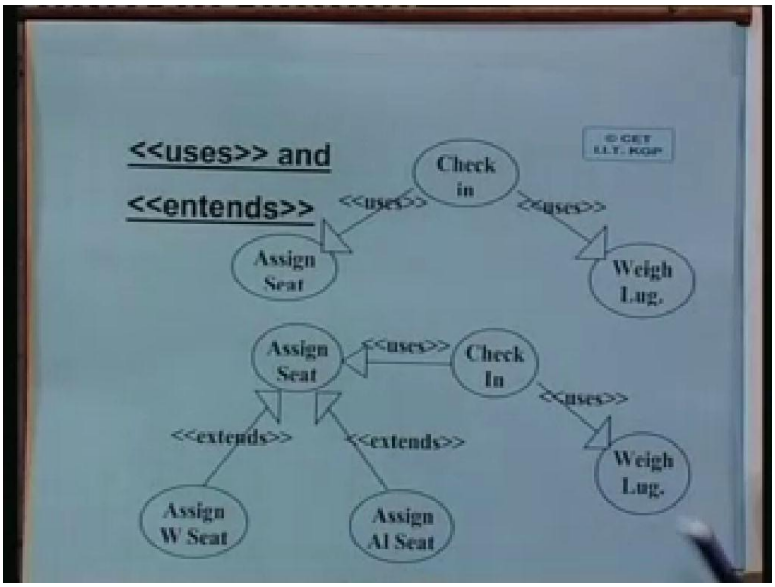
And the concepts of inheritance we have seen we have to identify early. Now, you see the use cases are the first diagramming tool, which we are using when you are doing object oriented analysis and design. You are trying to find out the business functions and from the business function you are trying to identify some basic constructs. Now, let us see one or two examples, of what exactly we were discussing. Let us take this simple example, first let us take this the reservation system.

(Refer Slide Time: 06:47)



Take this simple example, of reservation system. So, it is a very simple example, basically what happens in a reservation system you have a booking clerk. So, exactly when you reserve try to go for a reserving a ticket what exactly you do. So, essentially basically there are three keep three procedures a first one can be called as checking in. So, the you have to reach the booking clerk and the booking clerk has to see that whether you are basically, allowed to be reserved in that particular case or you may not be allowed for this particular reservation, then after that either you could be added to this or you could be cancelled so this is a simple thing.

(Refer Slide Time: 07:49)

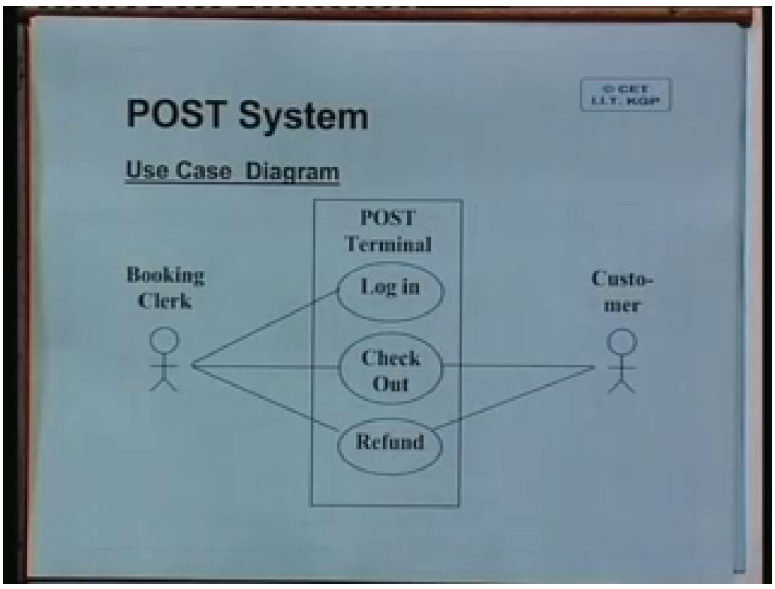


Now, let us see how the two things that are uses and entends we can have. So, when you have checked in, in the reservation system particularly let say in a near line then after you have checked in. The checking in process essentially, you know that you it is basically there are two things. One is that about your luggage weight age weighing the luggage and assigning the seat you have to assign a seat. Now, you can see that the assign seat further could be an window seat or it could be an al seat. So, you could be having two different kinds of seats either in the window or in the al. So, you can see that this is as shown the procedure or an assigning a window seat or assigning an a al kind of seat is by giving the extends relationship.

And you can have the uses so you see the check in is the use case that use case uses two different use cases weigh luggage and assign seat. The assign seat is basically the

extension of two other use cases. So, this is the very basic constructs that by using the uses and extends you can basically, identify a large number of different constructs, which can actually be possible is it clear basically. So, you have first drawn a very basic overview type of a diagram where you have simply shown the your a simple use case and actors then as you take those use cases and you go further. Then you may see that by using the uses extends or generalized depends etcetera. You can have lot of further constructs and you can give more and more details.

(Refer Slide Time: 09:56)



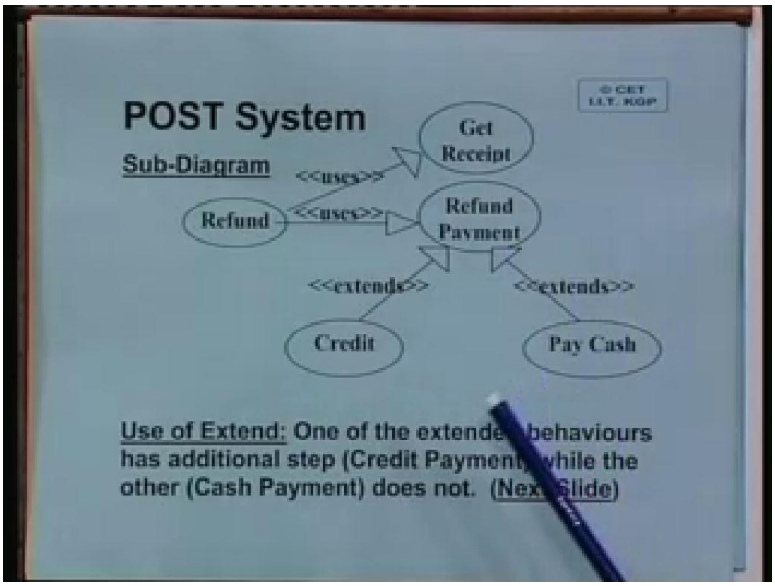
Let us look at these example, once again that is the post system basically, the post system or the point of sale terminal then have more than these only three basic processes are identified here we have the booking clerk and we have the customer. Now, what is happening in the post terminal we have three basic use cases, the booking clerk has to log in to the terminal and then when a customer arrives with it is items the customer has choosen then he has to check out procedure basically, the clerk what exactly the clerk does. The clerk has to see the items what are the different items and know make a bill and then the customer pays the money and that is the procedure that is see you have to try to look at it there could be lot of manual activities.

But the basic computerized activities could be that check all the items, one by one produce a bill and receive payments. Similarly, when it comes to the refund what could be the basic activities? You have to get the receipt to check that this is a real purchase

and then after getting the receipt make payments make payments. Now, when it comes to the payment you can make the payment by you know credit card or a by kind of a credit or you can pay cash. You can credit the money to the account of a customer if such a facilities available or you can simply pay cash. So, how it is to be done tell me what are the constructs you should use? Say particularly let us take the refund case.

So, we have the refund use case the refund use case is constituting of what other use cases yes one is that is later, but, before itself you have to get the receipt check the receipt or get the receipt. You know that receipt portion checking or getting the receipt validating the user basically and second is paying the user refund payment making the payment. So, how to show these two use cases because they are actually, part of the refund process we can make use of the uses or includes both are equivalent. And then since there are two kinds of a payment process the credit and the pay cash you can put it by using the extends see you can a just of a difference of these basically, both users similar symbol.

(Refer Slide Time: 13:19)



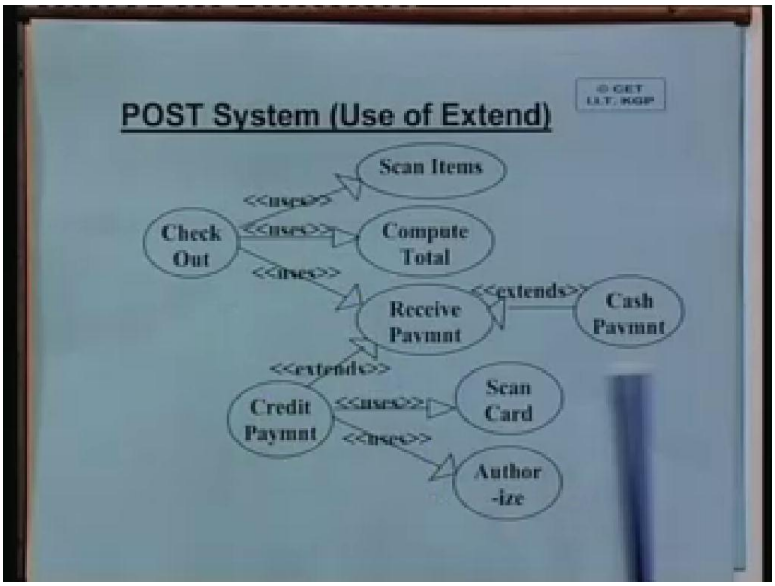
But the uses the arrow is from the use case to its constituents, look at this the refund has got two components get receipt and refund payment. So, we are having an arrow and these arrow uses you know get receipt refund payment. The arrow is from refund to the get receipt refund is from refund to refund payment because these are the two sub-use cases which are component of refund. But these two see the refund payment basically, is

of two types is it there are two types. So, when you are using extends when you are using extends basically, you are having the credit and pay cash credit and pay cash both are basically, part of the refund payment process in the sense that two different types of refund payment process.

So, the arrow has to be like you you recall your diagrams we are drawing the arrow from the sub-class to the super class. So, in this case also the arrow is from the component of the type to the original. So, the extends this way. So, this is how you show the further details in an use case. Now, use of extend one can ask what is the purpose of using this extend do I get some specific benefit out of these? The idea is one of the extended behavior has additional step suppose, the credit payment has got some additional steps what are those additional steps basically, may be a you have to scan the card. So, basically when you are crediting into; obviously, we are not talking about credit cards credit card basically, is used for deducting money out of it.

But suppose, we have some card in which credit is possible credit to your account then anyway you have to get the card the scan it and then authorize it until it is authorized it will not be used. So, one can say that two procedures are there scanning the card and authorizing it. So, this is how you can show these; obviously, for a refund it is not shown I mean since the refund you can show it in the refund also. I have not done it deliberately from refund the basic idea is since this idea of crediting in the refund is not very popular in our country, you do not know much about it. So, I have try to put it from the check out point of view look at this example.

(Refer Slide Time: 16:30)



We have the check out the checkout has got basically, three components. There are three different use cases which are part of the check out process one is this scan item. So, each item has come you have to scan each item listen or to read out suppose, the person in a super market has purchased thirty units or say twenty items. So, you have to take each item scan it is bar code and naturally, from the bar code in your computer you must have got the etcetera store and then a using all these things the second activity could be compute total. So, after you have computed total third is receive payment. Now, receive payment has got two extensions there are two types of payments, one could be cash payment another could be credit payment.

Payment could be made by the customer either through cash or through credit. Now, suppose if it is through cash fine, but, if it is credit then there are again two scan card and authorize. So, the credit payment uses scan card and also authorize. So, this is the basically, shows the power of extends see if the extends facility was not there, try to understand when it comes to the receive payment then what you see you do not know whether it is through credit or through cash. Then instead of one diagram you may have to draw two diagrams, one if it is you may have to write if it is credit payment then these diagram if it is cash payment then it is these diagram.

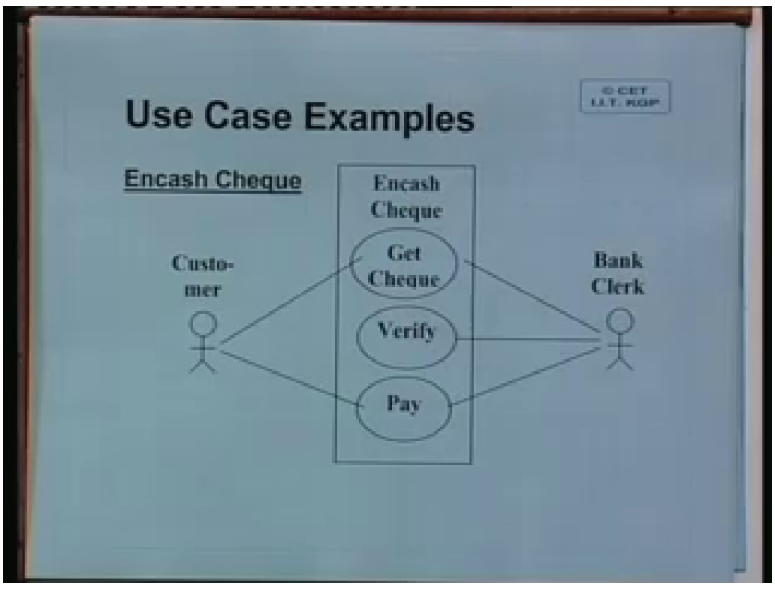
Because of the credit payment diagram you are to have processes likes scan card and authorize and in the cash payment do not have them. So, that is basically, the power of

your extends. So, you can show more details by use of extends not only that it helps you later as we shall see afterwards how all these can actually, help in revising further into our object oriented analysis. Now, let us take one more example that is about your encashing cheque in a bank. We see encashing cheque in a bank we have already done the function oriented analysis and design basically; we have drawn the dataflow diagrams for cheque and cashments.

So, recall that what exactly we were doing you present your cheque in the bank counter then someone will validate your cheque then it will validate your account validate your signature, gives you a token then you present the token to a cashier and then the cashier pays you. So, essentially this is how it happens in a bank now, whether the token is they are allot. What are the basic processes which are involved here? So, that will be our business use cases.

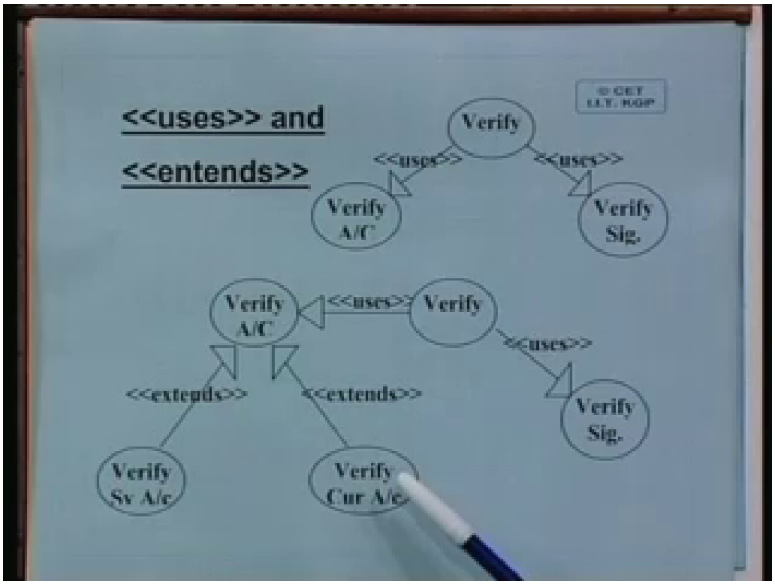
What are the basic things? Yes payments. So, basically, one is the getting the cheque getting the cheque second is verification a process of verification could be your signature verification, could be your account verification whether you have money in the account and finally, the payment process. So, this is the use case diagram for the thing that is customer and the bank clerk. You see the verification process as nothing to do with the customer the verification is done by the bank clerk.

(Refer Slide Time: 20:46)



So, you have the get cheque verify and pay these are the three basic use cases then encash a cheque. Now, see it does not end here please understand this you have to now, the use case descriptions. For example, get cheque what are the sequence of events what exactly happens? Verify what are the a basic processes? How do we verify? So, you have to give that in detail either as I said through narratives or through a scenarios right or in the questionnaire format then the payment. So, this is the basic processes of use case. Now, let us take the verify accounts the verify account how exactly we do verify a account or verify? What are the processes sub-processes of verification? look up the file to begin with you have the verifying the account and the verifying the signature these are the two basic processes.

(Refer slide Time: 22:19)



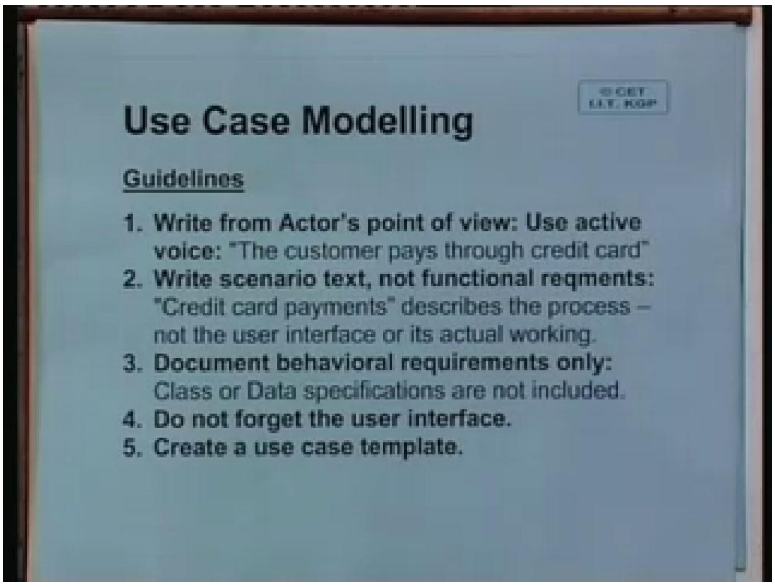
So, you can see that this is how uses and extend at least one simple example, that verify uses verify account or verify signature. See so, you please understand that uses means these are the two components whenever, you are verifying you have to verify account as well as signature the accounting and verifying account and verifying signature and not two different types of verify it is not either or kind able situation you have to do both. So, uses then again the account could be a savings account or it could be a current account.

The verification is slightly different for current account and savings account in savings account overdraft is not allowed in current account overdraft is allowed you can draw more cash sometimes, not all the all the time even if you do not have so much money in

the bank. Provided the bank has a guarantee that some money is going to be deposited quickly. So, that is how the two verification processes are different. So, we have the verify a savings account or you can have the verifying a current account. So, that is how we can go further, but, this is only one portion of that is the verify similarly, pay also you can have further details in the use cases.

So, essentially what we basically, do in an use case diagram we draw the use case diagram first in the just the actor and the use case and secondly, we try to use the concept of uses or includes and particularly extends by which we try to draw more details. But as I said again and again that it does not end here for every use case you must keep the sequence of events in written form. Now, some of the very important thing about the use case modeling some important guidelines.

(Refer Slide Time: 24:32)



First and foremost important outline guideline for use case modeling is that you must write from actors point of view use active voice that is the customer pays through credit card this is how not the payment is done by the customer through a credit card. So, as far as possible write from actors point of view and use active voice. Second one write scenario text not functional requirements. So, that means, when it comes to credit card payments describe the process not the user interface or its actual working. See basically, what it exactly means, that when you are making the credit card payments you basically,

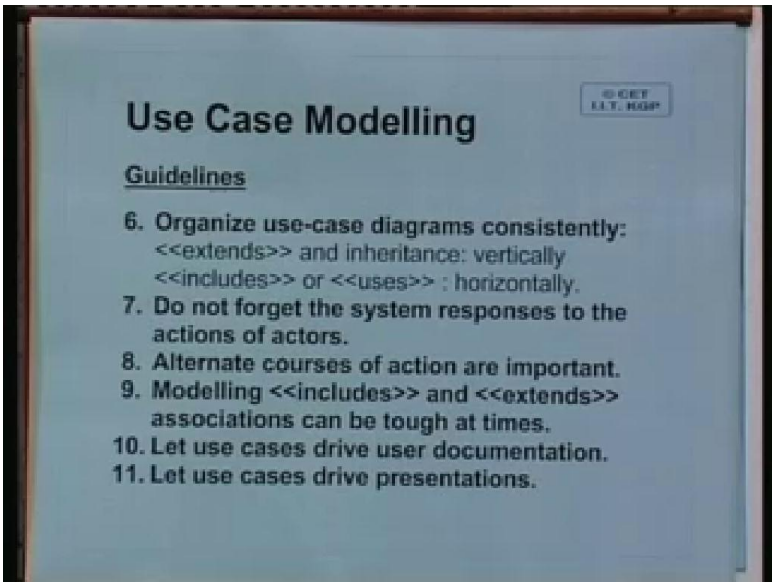
describe what you see. While your you know making the seeing the credit card payments from the users point of view; that means you have the credit card you scan the card.

Then you authorize the card that is basically, the bill that is generated the produce the bill and the bill has to be signed by the customer. This is the level at which you must confine yourself in the use case. Do not worry what happens inside how it is done when you actually have the credit card being scan then how this scanning takes place. What is the methodology of scan? What kind of user interface you must have? What kind of hardware and software you must have? So, there are many more technical questions which comes appear let us not bother the use case with the implementation details is it that is what it meant here.

The document behavior document the behavioral requirements only at this point there is no need to try to identify the class or the specification of the data. You just see what is happening do not forget the user interface, see when we talk about the do not worry about implementation, but, what is if suppose in some situations the user has to participate, do not over overlook it. That is suppose you have the a credit card or suppose, you have a customer who is logging in and the customer has to give a password and second time the customer has to authenticate once again.

So, do not forget this if you forget then the one very interaction with the actor is missed. Then create an use case template a template usually is because you have to write a number of use cases so if you have a template it makes it faster. Then organize the use case diagram consistently right say how to do it extends and inheritance in a vertical manner includes or uses for horizontally. That is a very simple thing to understand if you show it horizontally that is the uses or includes it makes, easy to read similarly, extends or inheritance in a vertical manner.

(Refer Slide Time:28:02)



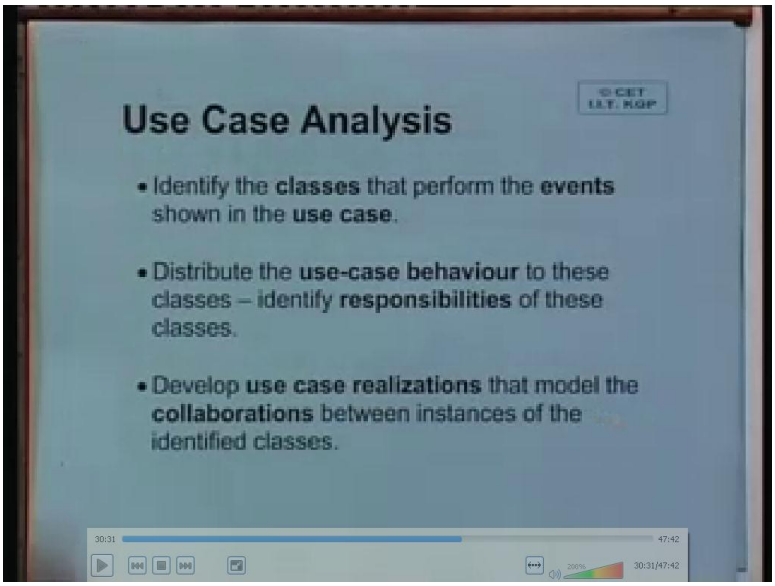
Then do not forget the system responses to the actions of actors. So, it is also very important. So, whenever an actor does something suppose a student is registering for a course and the student enters a form registration form. So, when a registration form is entered then the feedback that the system gives that also must be included alternate courses of actions are important. So, these means, that there are say the credit card payment is not the only payment there could be cash payments either that also to be identified. Then modeling includes and extends associations can be tough at times this is an important observation.

Sometimes a invoke a just for finding whether, it is includes or it is extends is it a type or it is not a type we are would be in fact, the example, basically says that in a real life situations many situations may arise where the people instead of going ahead with the analysis and design. There actually simply fighting whether they are should be more than one extend, extend should have three elements or four elements or two elements. So, there could be lot of debate here. So, if you spend all your time in this debate then you convert move ahead.

So, although the uses and extends are there basically, to show the thing in more detail end of the day you have to get the requirements that is the idea and the use case diagram is simply a coding of the requirements. So, do not spend endlessly on this. And let use cases drive user documentation; the user documentation can be coming out of the use

cases. Similarly let use cases drive presentation because the use case is a customers view is an world view business view. So, every presentation that you may be making to your customer to your client the use case could be a very important tool. So, this is about the use case now, let us move ahead with what is known as the use case analysis.

(Refer slide Time: 30:31)

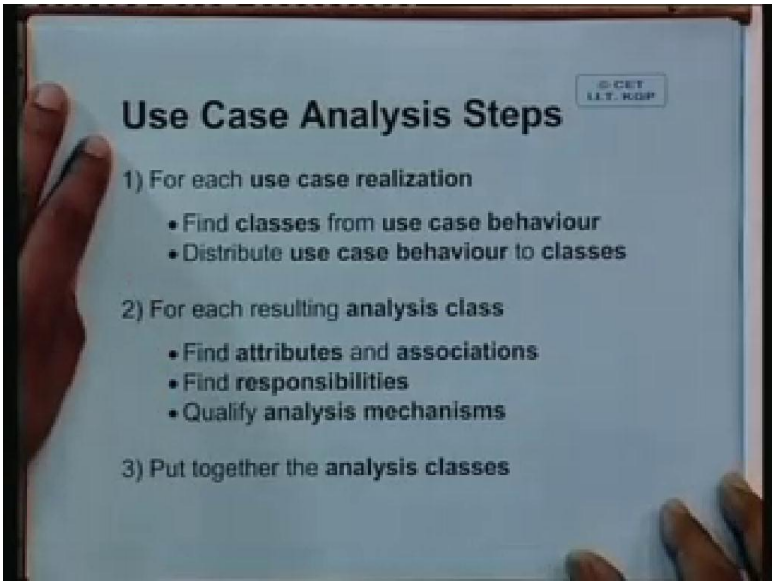


Now, we are going to analysis the use case actually is a you know level where we are identifying the requirements. And the requirements we are putting in a perspective now, how do we analyze it further. So, that we get what is known as use case realizations. The idea of use case realizations is that if you have to implement the use case in the analysis. Then how do you identify the classes, the end of the day any object oriented analysis, you have to identify the classes. You have to identify the classes for each class you have to identify with them attributes and operations. So, how the use case helps us in identifying the classes, that perform the events shown in the use case that has to be done.

Then distribute the use-case behavior to these classes; that means, an identify the responsibilities of these classes that is also very important. So, as I said that every class has got attributes and operations. So, this operation has to come up from the use case descriptions as is it not. That you have identified all these descriptions or behavior as is coded in the use case. How you can transfer that into the classes? And then the develop use case realizations that model the collaborations between instances of the identified classes.

So, basically not only that we know that the between the classes we have multiplicity. There are different kinds of associations between classes this is that one label and the operations label what are the collaborations, which class is collaborating with each other class and what are the responsibilities. So, all these things are vital for any object oriented analysis. So, that you can later on develop the class diagram.

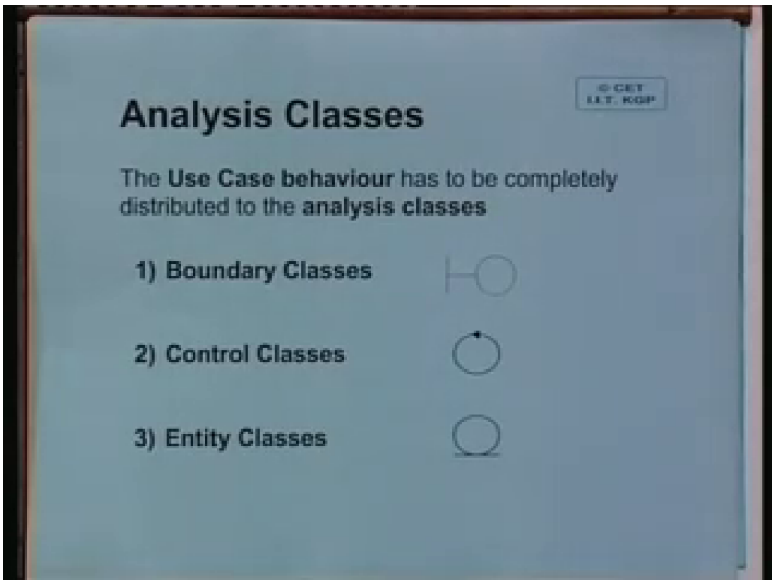
(Refer Slide Time: 32:45)



Now, for each use case realization what is required? Find classes from use case behavior distribute use case behavior to the classes. These already we have say for each resulting analysis class find attributes and associations find responsibilities and qualify analysis mechanisms. These are the steps and put together the analysis classes. So, we have to find all these. What are the different analysis classes that come out of the use cases and the responsibilities? And finally, the analysis mechanisms we will discuss all these things in detail one after another and how do we put together the analysis classes. So, this is the basic three steps of the use case analysis today move further.

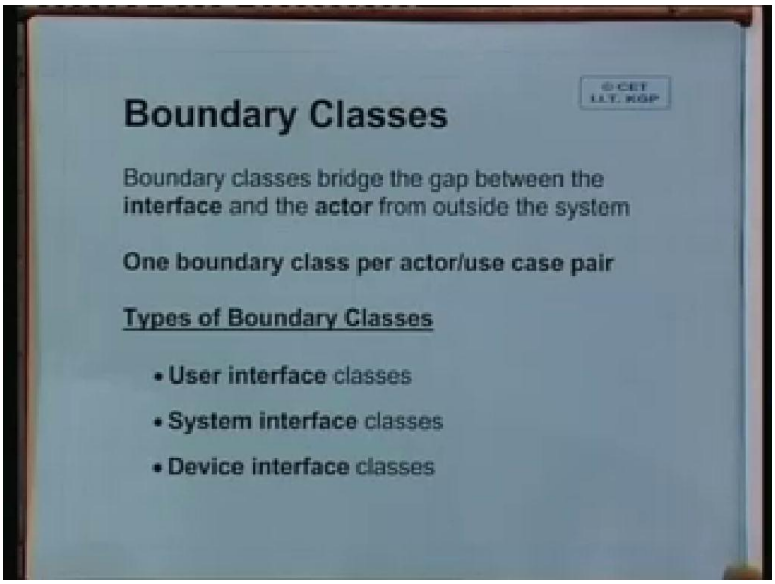
So, we have the each use case realizations we have to find classes, have to distribute the use case behavior to classes and for each analysis class we have to find attributes and associations responsibilities, and we have to qualify the analysis mechanisms. So, let move over.

(Refer Slide Time: 34:31)



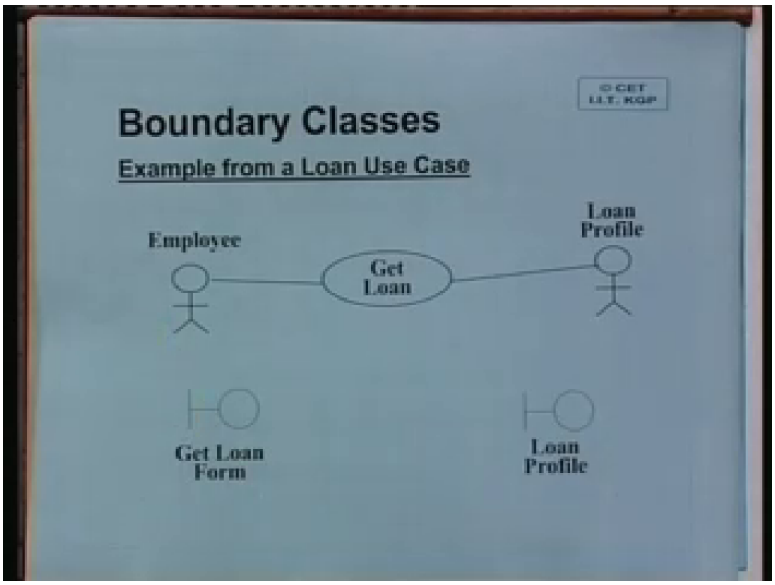
Now, there are three types of analysis classes. The use case behavior has to be completely distributed to these three types of analysis classes. What are these three types of analysis class? The first one is called the boundary classes the second one is the control classes and the third one entity classes. So, we have the three notations for a boundary class and the control class see that in the control class we have the circle and on the circle we have been we have put an arrow. So, circle with an arrow and with a circle with line below, it is the specification for the entity classes boundary class a circle and a t type of structure. So, this is the sign, which are basically to be used for boundary classes control classes and entity classes. Now, let us see a first one that is what is a boundary class?

(Refer Slide Time: 35:42)



What is a boundary class? The boundary class is bridge the gap between the interface and the actor from outside the system. And there will be one boundary class per actor use case pair. So, before we move for that let us try to see one such example of an boundary class.

(Refer Slide Time: 36:09)



See we have an employee and we have get loan use case and there is a loan profile. Now, why the loan profile should be an use case? Basic idea is that the there are different kinds of loan different kinds of loan with a very different details say for example, you may take

an a one lakh fifty loan pay it over five years or pay it over ten years pay it over fifteen years. So, you may also take two lakhs, three lakhs, four lakhs all the way up to fifteen lakhs all the and you can pay it back in different number of years. So, depending on how much you have taken and how many years you will be paying it back the interest rate may be again decided accordingly.

Similarly the interest rate may itself be varying sometimes, what happens people go for a fixed interest rate; that means, whatever happens the interest rate will remain let say around ten percent or someone may say dynamic interest rate. As the market changes because as you can see that housing loan interest rates are falling may be sometime later it may rise again. So, when it is falling you are very happy that we have gone for dynamic interest rates, but, if it then it will not be very right whatever it is. So, there could be large number of different kinds of loans that one can take this is what we are calling as loan profile.

So, when in imply once to get a loan years to one from all these different loan profile. So, that is the only simple use case we are talking about an; obviously, all they are the verification process and also on and so forth. So, now, you see when you want to implement this try to understand we have the employee and the loan profile and the get loan. When you have to implement these, particular situation? Then you must have an interface by using which the employee will enter his details at some level of the other either if you think of a batch kind of an application then some clerk may be entering this loan profile a loan form.

That what is his name number employee number name then address then basic pay eligibility of a loan etcetera. Whether he has taken any previous loan if he has taken a previous loan has it been adjusted so all these a details are to be entered. And so this is like an interface. So, this interface has to be there similarly, on the loan profile side you may also think of a boundary class apart from the entity. So, the entity is an employee it does not mean that just because of a get loan form and have taken employee details the entity vanishes entity remains.

So, we have the basic entity of the actor that is employee. And we have the get loan form and the loan profile these are our boundary classes. Easy concept of boundary class clear that is why what I said is that boundary classes bridge the gap between the interface and

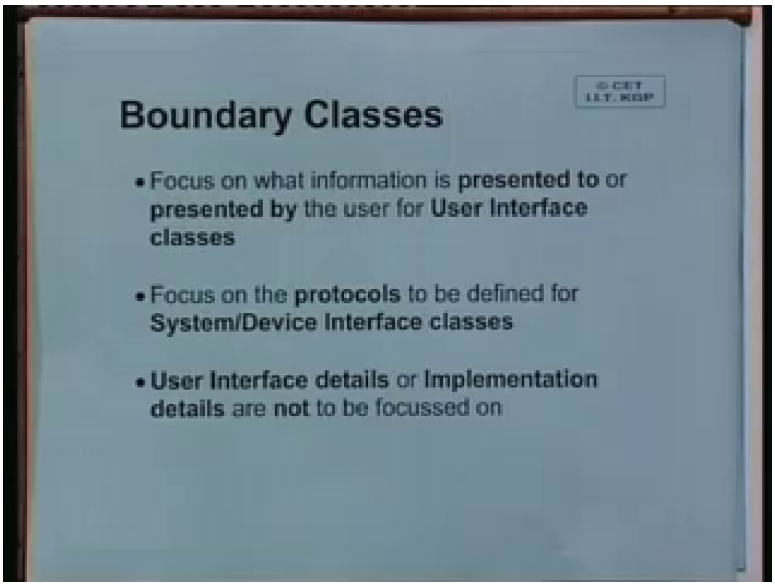
the actor from outside the system and there is one boundary class per actor use case pair per actor use case pair. So, we have the between employee and get loan there will be one boundary class. Between get loan and loan profile there will be another boundary class. So, that is the idea of the boundary class.

Now, there could be three types of boundary classes we can have an user interface classes for example, between an employee and the get loan. So, employee has to interface the student and the course registration. The student there has to be a form through which the student enters the course. There could be system interface classes system interface I have given you an example, earlier suppose you have a basic financial accounting system where, you already have a legacy system in payroll if you recall. I have given that example that you want to implement the new system the new system can only be implemented only after the you know the remaining things are develop, but, you may continue to use the payroll as a legacy system.

Until analyze the payroll of the new system is completely develop verified, validate only then you want to move over. Till such time the old payroll system is an actor it has an interface with the system. So, this is a kind of system interface. So, in a banking system when you are having an authentication process suppose, an remote customer wants to pay remote customer wants to pay online. So, before he begins the process he has to log in. So, he keeps in his terminal the password confirmative the password and all that.

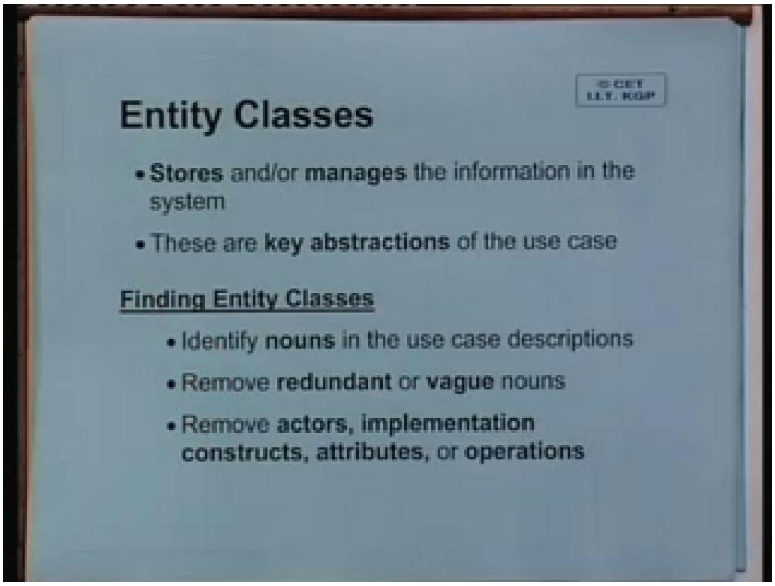
But that does not make at he is automatically logged on, no the system has to authenticate him. There has to be a system interface somewhere. So, that is the kind of system interface sometimes it can even be an device interface say there could be a specific type of device we preach your basically interacting. So, suppose in a particular case if a computer basically, acts the computer is preprogram and the computer actually, acts as a interface to a interact with your use case. So, you can have all these three different kinds of interfaces.

(Refer Slide Time: 43:10)



Than focus on what information is presented to or presented by the user for the user interface classes. So, essentially for an user interface the basic idea is the focuses on the user interface and the information that is being presented to or from or by. The focus on the protocols to be defined for system or device interface classes whenever, you have the system or a device interface you must have a protocol define similarly, the user interface details or implementation details are not to be focused on at this stage. At this stage you simply represent it as a form of boundary class. The next important thing is a what is known as the entity class.

(Refer Slide Time: 44:12)

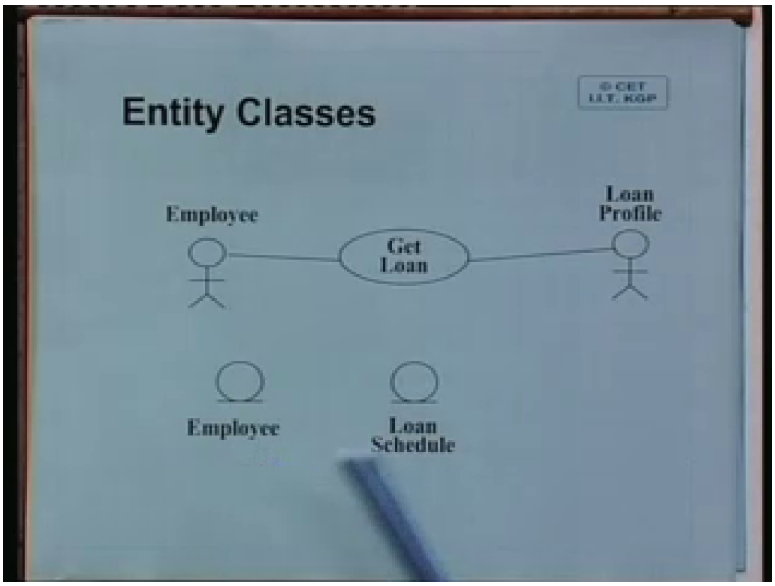


The entity class is a very important concept In fact; most important in this particular scenario that is a entity class it stores and or manages, the information in the system. So, basically if you are going to do a database design in this particular system then what are the constructs that you would have chosen? So, all these constructs what you would have choosen are going to be part and of the entity class. So, these are the key abstractions of the use case you recall I told very specifically, when we are talking about a database modeling a very first class. I said that the entity relationship diagram is an abstract diagram is a data abstraction is a kind of data abstraction.

If the data is same definition of the data is same wherever you would design the database it will be one and the same cannot be two different things. So, a finding entity classes you have to identify nouns this is the booch method of finding classes. Basically identify nouns in the use case descriptions remove redundant or see this is where the use case description comes in remove redundant or vague nouns remove the actors, implementations, constructs, attributes or operations. So, you should see that it should be an implementation, which is not just an actor name just because the, somebody is an actor he did not be an entity.

But that does not mean no actor would be an entity. The basic idea is whether we are trying to collect information about that particular thing concept thing or person if it is so yes. Say for example, a booking clerk in a post terminal in a post point of sale terminal, tell me the booking clerk is the going to be an entity yes or no, no is an actor the booking clerk is operating the system, but, the system does not need any information about the booking clerk whereas, in this particular example, between employee and get loan tell me can employee be an entity yes or no has to be because we are collecting data about this employee. So, we will just close it for today this is the diagram which shows that two entities in this particular case.

(Refer Slide Time: 47:01)



We have the employee we also, have the loan schedule and we want to keep information about the employee and also the loan schedule. So, there may be more than this, but, on a preliminary scan we find two entity classes in this particular diagram. So, we leave it here in our next class we shall see further, what is a control class and how the boundary class the entity class and control class at put together. So, that is all.