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

Lecture - 32
Factoring Use Cases

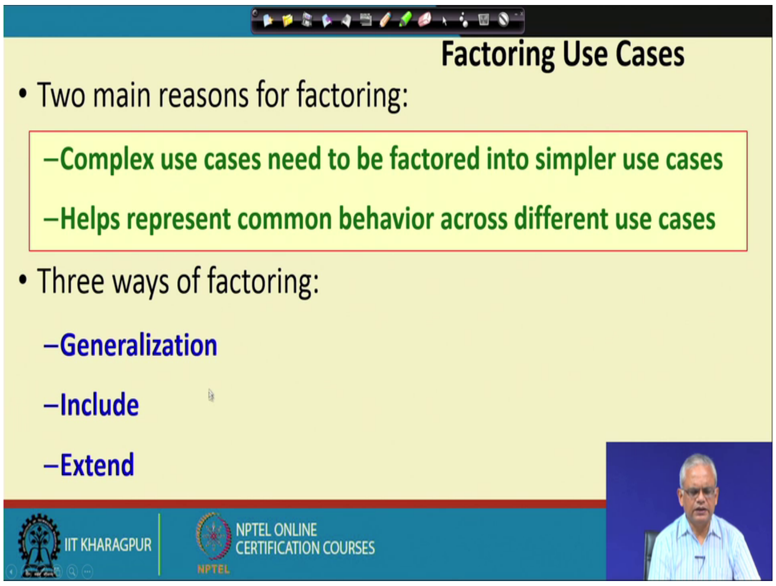
In the last lecture, we had looked at use case modeling said that that is the very basic fundamental model which is constructed first in any development and it captures the users view. We saw that it was really the high level requirements that are called as the use cases, but then here we have a graphical modeling of the use cases and that will be accompanied by text description.

We, had discussed about graphical model of use case what do the different elements indicate. And, given a use case model what do we interpret with the like we see the use case model of a system somebody showed us, can you understand what does this use case model mean?

In this lecture we will see more details of the use case modeling namely, given a high level requirement sometimes the high level requirement or use case is complex very complex some are very simple. So, can we decompose this complex use cases and represent them as sub use cases. And, also we will see how to write text description and finally, we will see that how do we identify the use cases for a given problem? And, how to develop this use case model, because that is the ultimate objective given a problem we should be able to draw it is use case model.

Now, let us see how do we factor a use case? First let us understand why we need to factor? There are 2 main reasons why we need to factor a use case?

(Refer Slide Time: 02:21)



Factoring Use Cases

- Two main reasons for factoring:
  - –Complex use cases need to be factored into simpler use cases
  - –Helps represent common behavior across different use cases
- Three ways of factoring:
  - –Generalization
  - –Include
  - –Extend

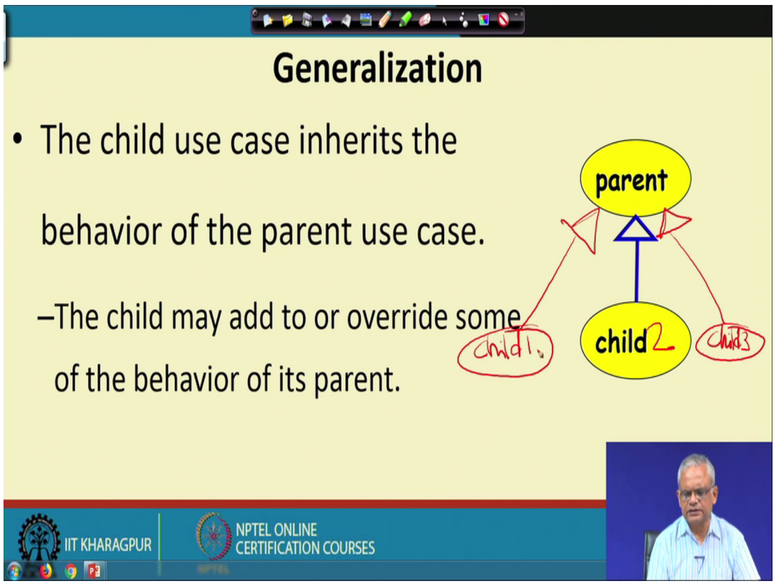
The first is as we proceed with design activities we will see that unless we factor the use cases and make it into simpler use cases. The design process becomes extremely complicated, because we will develop some diagrams let me just name now the sequence diagram and so on.

Which will develop based on the use case diagram unless the use cases are simple, the sequence diagrams etcetera will become extremely complicated and at that time we would have to again come back and relook at our use case model and simplify, but then it is a good idea that while developing the use case model, we do not represent complex use cases we try to factor them or decompose them into simpler use cases.

The, second reason why we need to factor is that some aspect of one use case may be common to another use case. If, you do not identify them here in the use case model, latter there may be a duplication of the design and code. So, if there are some part of the functionality, that are similar across to use cases. We need to identify those similar functionalities and represent them here in the use case model. So, that we develop them only once. It helps us to reduce the design coding effort and identify what are the common functionalities across the different use cases.

So, these are the 2 main reasons; why we need to factor use cases, but then the next question comes how do we factor? We will use 3 techniques for factoring; one is called as generalization specialization. The second technique is called as include the third technique is called as extend. So, let us see these 3 techniques for factoring a use case.

(Refer Slide Time: 04:52)



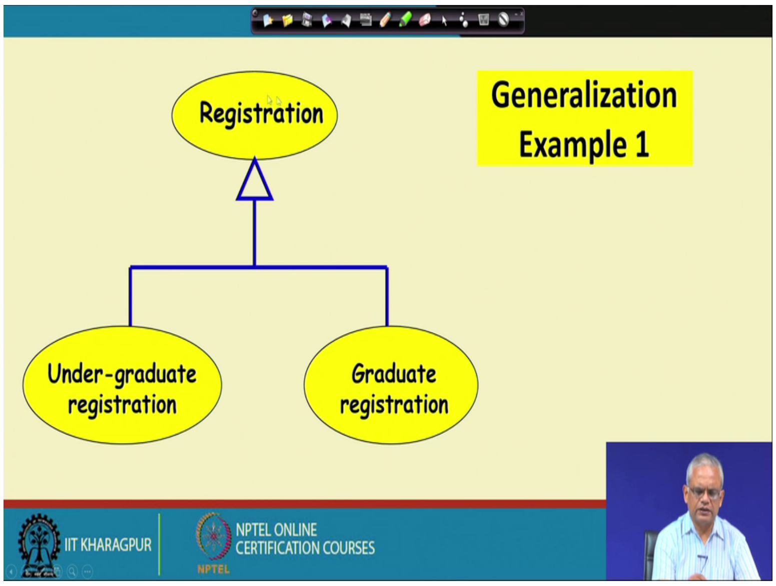
Generalization

- The child use case inherits the behavior of the parent use case.
  - The child may add to or override some of the behavior of its parent.

First let us look at the generalization here; there are some functionality, which is common to many other use cases, but then there are some minor differences. Then we show the common functionalities as the child use case or the base use case and this is the child use case, which has this functionality plus some extra functionality. So, we can have different child use cases. This is the common functionality, the parent is a common functionality; child 1, child 2, child 3 they have these common functionality, but something extra.

This helps because, we do the use case here parent use case we do the design code etcetera and then use that slightly extend that add some more functionality for these different use cases. If we do not do this, then we write the 3 use cases separately then we keep developing each of this and there is a duplication of effort and also these may become large. Here the parent once we develop there are only small part in the child that we need to develop.

(Refer Slide Time: 06:52)

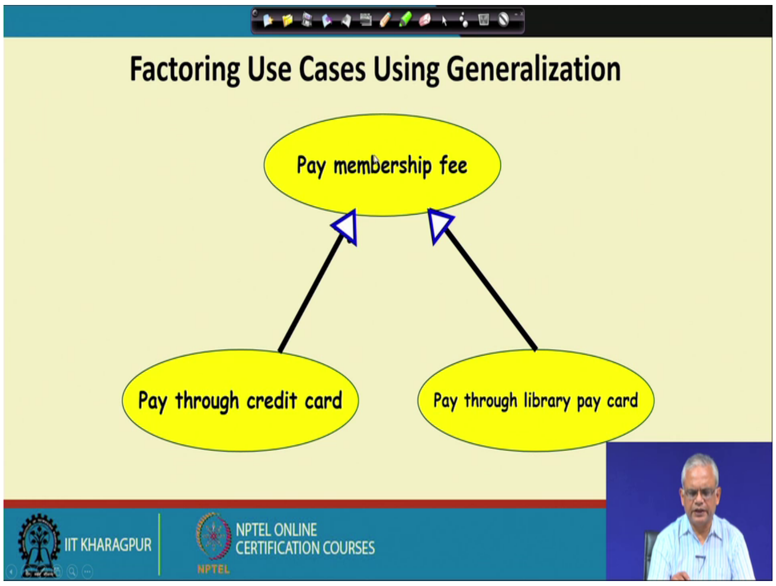


One example of generalization let us say student registration system, before the semester starts the students need to register. And, we have 2 types of use cases available one is called as undergraduate registration and the graduate registration. In the undergraduate registration let us say the fee needs to be paid before the registration can start. Whereas, in the undergraduate registration let us say there is a option that the fee can be adjusted from the scholarship, because all graduated students they get some scholarship in this specific institute for which we are developing this diagram.

And, let us assume that registration is more or less same they need to enter the semester the hostel accommodation etcetera etcetera, but there is a minor difference between a undergraduate registration graduate registration, in undergraduate registration the registration would not be complete until the fee is paid whereas, in a graduate registration a question will be asked whether you will pay the fee now or you would like the fee to be adjusted from scholarship.

So, we do the registration part which is common to both of this, and then the fee payment here and you are asking the question that whether, it will be adjusted or fee payment. A small change in behavior will be implemented in these 2 use cases and the registration use case the basic functionalities of the registration will be implemented.

(Refer Slide Time: 08:46)


Factoring Use Cases Using Generalization

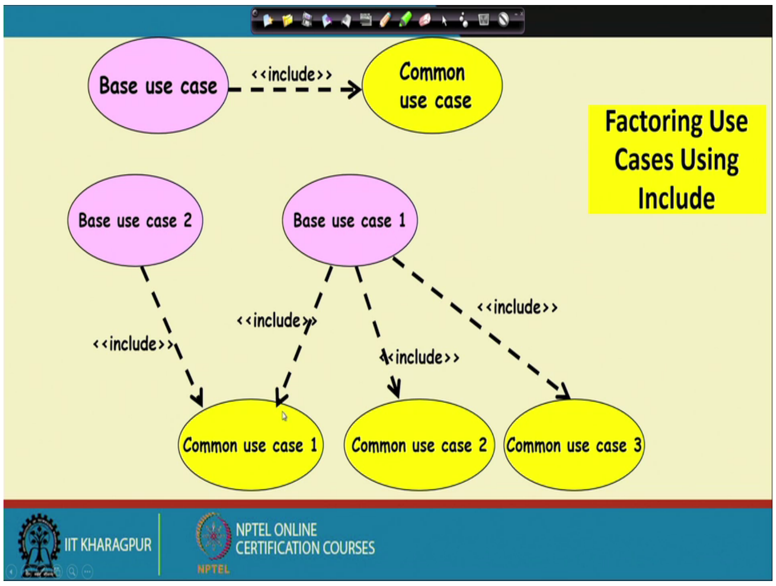
Pay membership fee
Pay through credit card
Pay through library pay card
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

There is another example here let us say a library software. Where we have a pay membership fee use case actually there are 2 use cases here supported by the software; one is called as pay membership fee through credit card and pay membership through pay card. The library issues pay cards where the members can deposit money in the pay card or they can they can pay through their credit card.

Now, most of the functionality are same for pay membership they would have to write their membership number see that there are no outstanding books and there are no complaints from the librarian and so on. So, this is the part that takes care of those functionalities and this 2 small differences between the pay through credit card and pay through library pay card are taken care through the generalization relationship, the generalization is a field is a arrow here diamond sorry arrow here this shape and a solid line.

So, these are 2 use cases, but they have common behavior there and we call this as a child use case, which are derived from the parent use case.

(Refer Slide Time: 10:29)

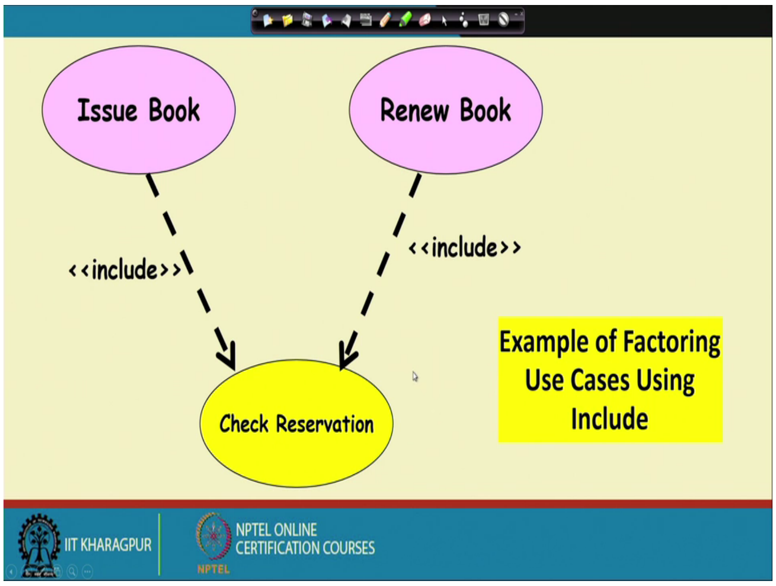


The second way to decompose a use case is by using the include relation. Here the base use case includes some use case, which is common across different functions different use cases. So, this is the representation dotted arrow and we write the stereotype here include. For example, let us say between 2 use cases we have this common use case one that is used across these 2 use cases.

In case of a generalization specialization there are many aspects which are common in the base use case and then these are extended in minor ways in the child use case. Here there are small parts which are common across these 2 use cases and we identify that is a common use case and write here include. So, it means that there are some functionalities small part of functionalities between the base use case 1 and base use case 2 and we can develop this once and then this will invoke this and this will invoke this.

So, include is it compulsorily includes this behavior as well. And, if it is a large use case we can it also helps us to split it into simpler use cases. We develop this separately develop this separately and therefore, the base use case becomes simple.

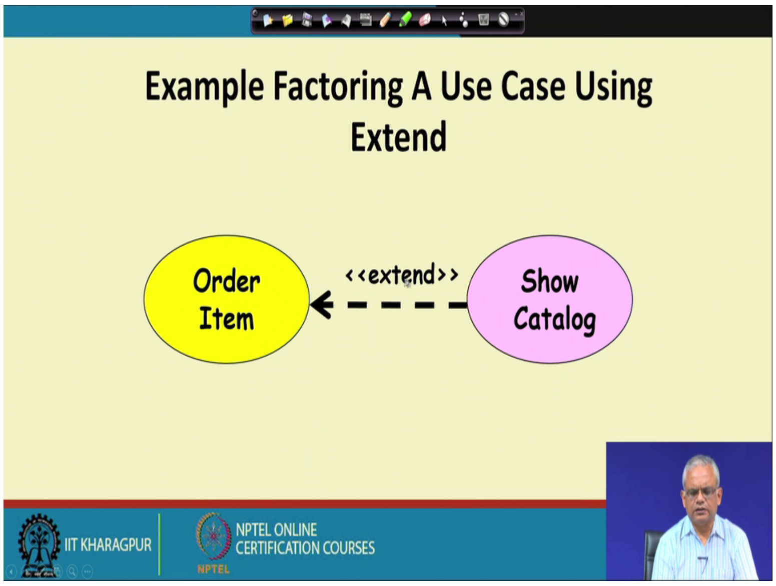
(Refer Slide Time: 12:28)


Issue Book
Renew Book
<<include>>
<<include>>
Check Reservation
Example of Factoring Use Cases Using Include
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

This is an example of factoring using the include relation between use cases. Let us say we have 2 use cases by the library information system which is issue book and renew book, but before a book can be issued let us say the functionality requires that we needs to check if somebody has not reserved that book.

Similarly, before a book can be renewed needs to be checked whether somebody has reserved that book, then we can just write this representation, it means that both these use cases require checking the reservation and this said check reservation functionality will be developed. Once we will have a separate sequence diagram code for check reservation and that will be invoked by both the issue book and renew book. So, it gets developed only once, it also helps to simplify these 2 use cases the issue book and renew book.

(Refer Slide Time: 13:49)



Example Factoring A Use Case Using Extend

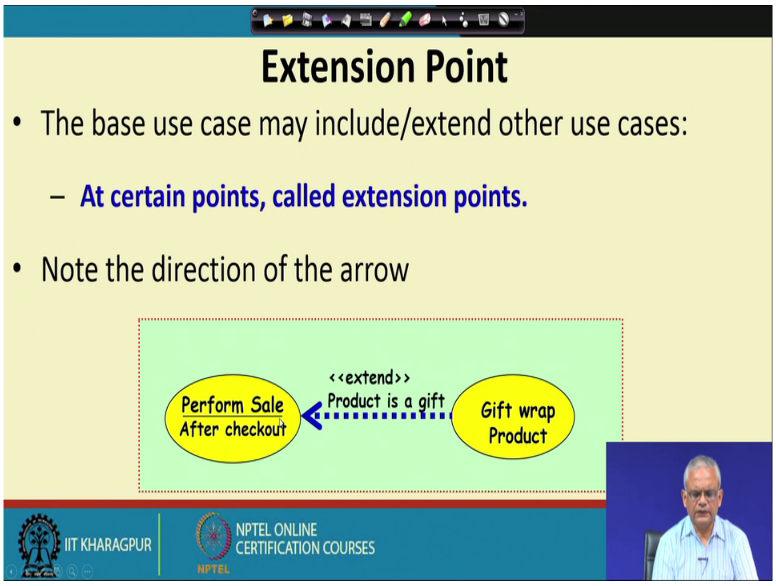
This is the third way to factor a use case called as the extend relationship between 2 use cases. Here, this is the base use case and it is extended by the show catalog. Let us say we are ordering an item on a e commerce portal.

And, when we order the item if we are sure about the item that we are ordering, we just go on entering the details of the item, but sometimes we do not know the item, we want to see the details of the item, what is the item code? What is the let us say we want to check whether warranty is given for how long and so on. Before, we can complete the order then we would as part of the order item. We may request a show catalog. And, in the catalog we can check, what are the different types of products that are available which you can order, what are the specification warranty and so on?

If, you are receiver of the product we know all the details we just do not invoke the show catalog we just keep on ordering. So, this is optional if you think of it, if there is extend relationship. Then, order item can execute as it is some user may execute the order item as it is, but some users may take help of this use case as part of execution of the order item they will request show catalog.

So, the behavior of the order item is is optionally extended by the show catalog, but just note the direction of the arrow here when it was include it was compulsorily includes the other functionalities. Here, it is optional the stereotype is extend and the direction of the arrow is in this direction whereas, include was in the other direction.

(Refer Slide Time: 16:17)



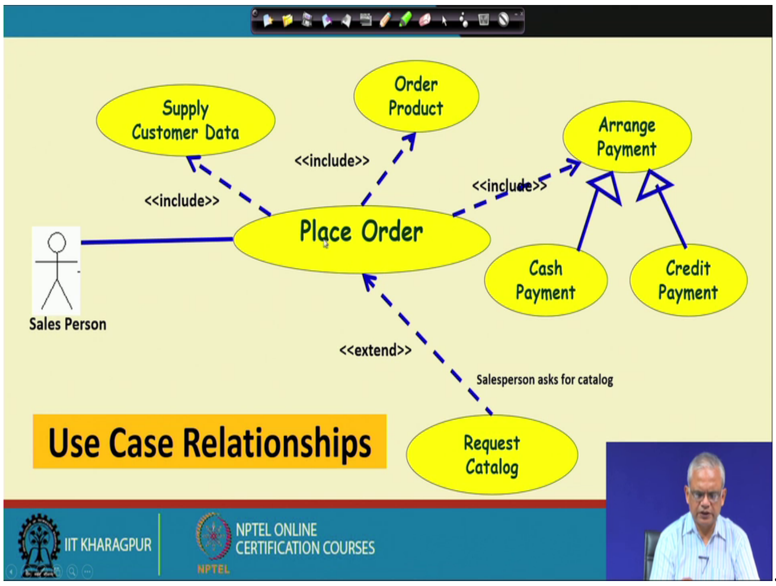
Extension Point

- The base use case may include/extend other use cases:
  - **At certain points, called extension points.**
- Note the direction of the arrow

UML also provides the facility to define a extension point, it helps sometimes to say that at what point in the execution of the order or perform sale an extend use case can be executed. Let us say during a sale somebody can request that the product is gift wrapped.

So, by this notation we can say that after checkout there will be an option that whether to gift wrap the product or not. So, somebody can invoke the gift wrapped product after checkout. So, that is what it means? So, this is called as the extension point at which this functionality can be invoked, but the extension point is optional we might not show in our design, in our model at what point the product the gift wrap product will be invoked.

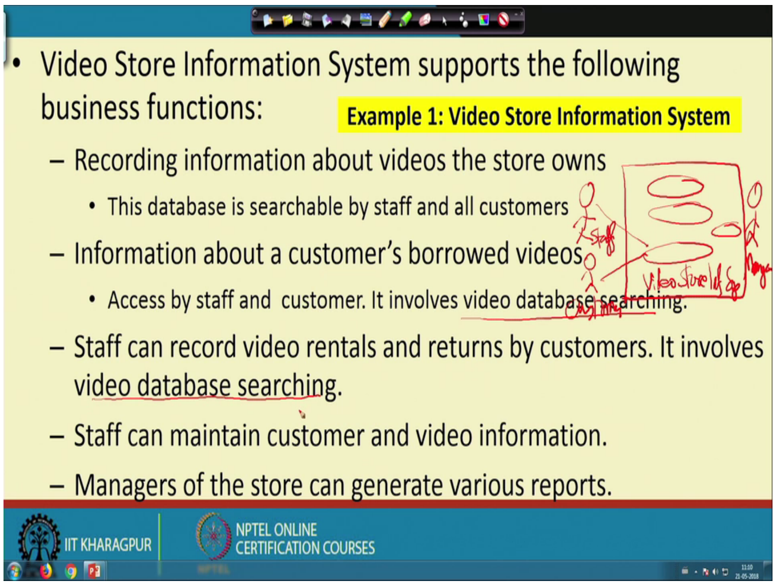
(Refer Slide Time: 17:32)



Given a use case model let us try to understand, that what does it imply? Here the sales person can invoke the place order use case.

And, the place order use case, while execution includes the supply customer data order product and also arrange payment. And, arrange payment are actually 2 types; one is cash payment and credit payment, and also while placing the order the sales person can invoke request catalog. So, this is extend and just see the direction of the arrow here it is in this direction towards the base use case, but here these are the include and this direction is different from this direction.

(Refer Slide Time: 18:36)



- Video Store Information System supports the following business functions:

Example 1: Video Store Information System

- Recording information about videos the store owns
  - This database is searchable by staff and all customers
- Information about a customer's borrowed videos
  - Access by staff and customer. It involves video database searching.
- Staff can record video rentals and returns by customers. It involves video database searching.
- Staff can maintain customer and video information.
- Managers of the store can generate various reports.

Now, let us see few examples how to develop a use case diagram? Let us take an example of a video store information system. And, it is given to us that it has the following functionalities. The first functionality is that all the videos that the store owns needs to be recorded and once these are recorded this is can be searched by staff and also the customer.

So, the available videos that the store owns can be searched by both staff and customers. And, the information about a customer's borrowed videos can be accessed by staff and customer. And, also it involves the video database search where the customers borrowed information is recorded. The third functionality is that the staff can record the video rentals and also the return of rented video by the customers.

Again this involves database searching video database searching. And, the staff can maintain the customer records, they can add customer delete customer and also they can maintain about the video information that is there should be able to record the information about the video they can enter the data about the video and also if some video is lost damaged etcetera there should be able to delete it.

The manager of the store can generate various types of reports. Now, how do we go about developing the use case model for this software? One is that we identify what are the functionalities and who are the users? We, can draw the system boundary and then we represent the users and then find out what are the functionalities they can represent they can invoke and represent those functionalities. As, you can see here we draw the boundary write video store information system and then draw the stick icon corresponding to different types of users.

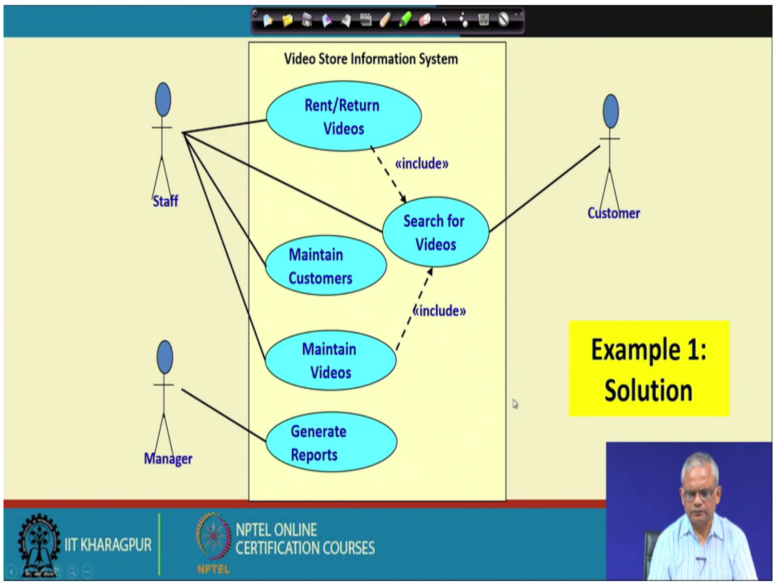
So, who are the users here? One is about staff and the customer. The staff and customer can query information about a customer's borrowed videos. The staff can record video rentals and also returns by customer and as part of this it invokes video database searching. The staff maintain the customer and video information the manager can generate various reports manager of the store can generate various reports.

So, we can see here clearly that there are 3 categories of users; one category of user is the staff.

Another, category of the user is the customer and the third category of the user is the manager. And, then having done that we can find out what are the functionalities invoked by each of this. For example, the staff we can find out that the staff can maintain customer and video information. So, you can write here maintain customer information, maintain video information, the staff can record video rentals and returns record rentals and returns. And, then the customer can query about the borrowed videos. So, the customer write can query about borrowed videos. So, is the staff, but one thing is that all this involved video database searching see here, that the staff rental recording the rental and return by the customer involves video database searching. So, this appears like a sub function which is included.

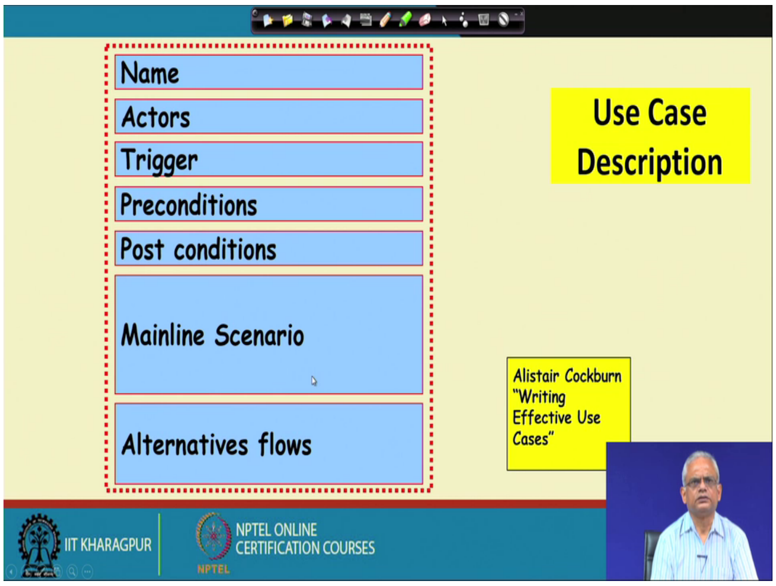
Across different use cases, here the video database searching is also required as part of the query information about the borrowed videos. Now, let us draw a neat diagram, we approximately know how to go about drawing the diagram let us draw a neat diagram.

(Refer Slide Time: 24:35)



So, the customer the staff and the manager are the 3 categories of users, the staff they interact with the rent and return videos use case, they maintain customer and maintain video records and the manager can only generate reports the customer can search for videos. And, see here that the maintain video, rent and return video they also use the search for videos.

(Refer Slide Time: 25:21)


Use Case Description

Name

Actors

Trigger

Preconditions

Post conditions

Mainline Scenario

Alternatives flows

Alistair Cockburn "Writing Effective Use Cases"

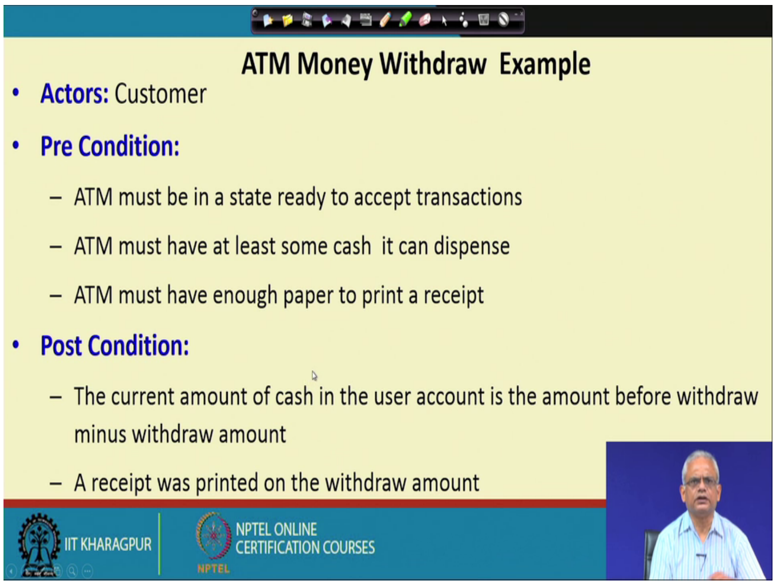
Developing the graphical model of a use case is the first step, but then that does not give us lot of details it just tells us, what are the requirements? What are the use cases and so on?

But, we need to also document the use cases there is no standard format that is prescribed by the UML, but then there are some way to document the use cases, which are considered as good practice. This is the one recommended by Alistair Cockburn in his book writing effective use cases we can follow that, but then there is no hard and fast rule that we need to document in this way.

We need to write the name of the use case. So, this is the text description that should accompany every use case diagram. Just giving the graphical use case model is not enough that does not portray or give lot of information the detailed information are given by a text description. Here, we need to write the name of the use case who are the actors of the use case these are also present in the graphical model. We need to write what are the triggers like what the different users external systems need to do to invoke the use cases the precondition and post condition.

What should be satisfied before a use case can be executed and what must hold true after a use case has completed? And, we will look at the scenarios in a use case and we will represent what is called as a mainline scenario and the alternate scenarios?

(Refer Slide Time: 27:18)



ATM Money Withdraw Example

- **Actors:** Customer
- **Pre Condition:**
  - ATM must be in a state ready to accept transactions
  - ATM must have at least some cash it can dispense
  - ATM must have enough paper to print a receipt
- **Post Condition:**
  - The current amount of cash in the user account is the amount before withdraw minus withdraw amount
  - A receipt was printed on the withdraw amount

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

We will take some sample problems, we will try to document in this manner this is a good way to document use case, but it is not a compulsory that we need to document always like this. We can deviate it from little bit the companies they have their internal standards, where they might enforce the same one or they might have small deviations from this, but this is more or less the way the use case documentation is done..

We are almost at the end of this lecture in the next lecture we will see how to document a use case we will take some examples and see how the documentation of the use cases can be developed?

Thank you.