

Object-Oriented System Development using UML, Java and Patterns
Professor Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur
Introduction Use Case Modeling
Lecture 03

Welcome to this session.

In the last lecture we have discussed about UML. UML is a modeling language used to document object-oriented designs, and it has become a standard. We have also discussed the different views that can be documented by UML, the different diagrams supported by UML using which the views can be documented and so on.

Now let's continue from that point onwards.

(Refer Slide Time: 00:49)

**Some
Insights on
Using UML**

- “UML is a large and growing beast, but you don’t need all of it in every problem you solve...” **Martin Fowler**
- “...when learning the UML, you need to be aware that certain constructs and notations are only helpful in detailed design while others are useful in requirements analysis ...”
Brian Henderson-Sellers

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | NPTEL

22

Before we discuss the different diagrams, let's get some insights using UML. One of the prominent authors in this area, in the language of Martin Fowler, “UML is a large and growing beast, but you don't need all of it in every problem you solve.” The statement is very true, even though UML has many diagrams, and it has many notations but then only a few of the notations are used all the time, and the others are specialized, used depending on the requirement. Another author Brian Henderson-Sellers says “When learning the UML, you need to be aware that certain constructs and notations are only helpful in detailed design while others are useful in requirements analysis.” This is also true. We will see that the requirements analysis activity is initially done where the requirements are elaborated in the

form of Analysis Model. We will see that some of the notations are used during analysis and some other notations are used in the detailed design. As we proceed in this course, we will realize what Brian Henderson-Sellers remarked.

(Refer Slide Time: 02:39)

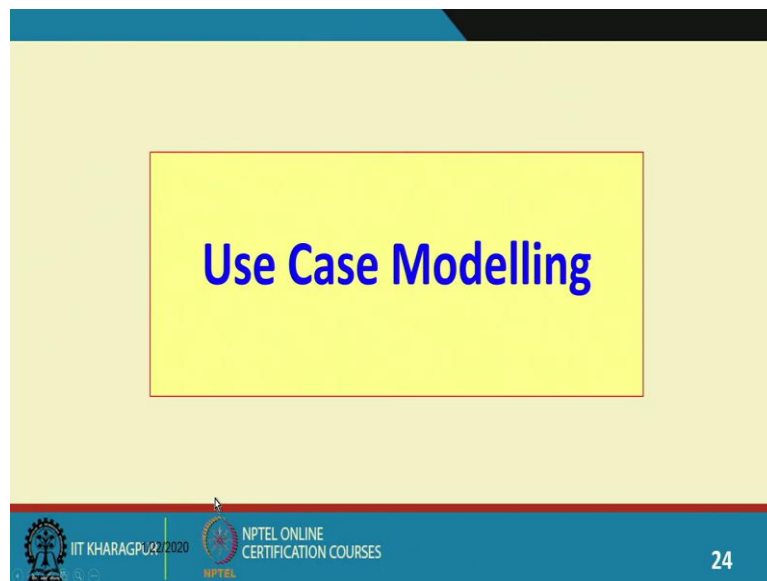
NO Are All Views Required for Developing A Typical System?

- **For a simple system:**
 - Use case diagram, class diagram and one of the interaction diagrams only.
- **State chart diagram:**
 - When class has significant states.
 - When states are only one or two, state chart model becomes trivial.
- **Deployment diagram:**
 - When system has many hardware components.

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES 23

To elaborate that, we will see some example. In the previous lecture, we have discussed five views, but are all the views required during the development of a typical system? The answer is no. For a simple system, let say student assignments, for that only the use case diagram, class diagram, and one of the interaction diagrams are sufficient. For some problems, we will need a state chart diagram where we model the states of the system. For a class in an application that has a significant number of states and transitions, we will need to use the state chart diagram. State chart diagram is a commonly used diagram after these three diagrams we can use: use case diagram, class diagram, and interaction diagram. However, we must remember that if a class has only one state or one or two states and so on, then the diagram becomes trivial. In this type of case, state chart diagram is not going to help. We would need the deployment diagram when the system has a number of hardware components, and the different parts of the software are installed on the different hardware. So, we can say that the deployment diagram is useful when our system has a number of hardware components, and different parts of our program are deployed on different hardware components. Using the deployment diagram, we represent which hardware component hosts which part of the code.

(Refer Slide Time: 04:56)



Now let us start discussing about the Use Case Diagram.

Use case diagram is the central diagram which captures the user's view. Whenever a system is developed, we first capture the user's view (what the user wants), then we document it and based on the user requirements we start development. We also get these diagrams verified by the user and therefore these diagrams are to be extremely simple because users otherwise can't understand it. So, the Use Case Modeling is simple set of notations, diagrams intuitive but then the caution is that we have to do it carefully because any mistake in Use Case Modelling will reflect in the other models because the other models are developed based on the Use Case Model.

(Refer Slide Time: 06:07)

Use Case Model

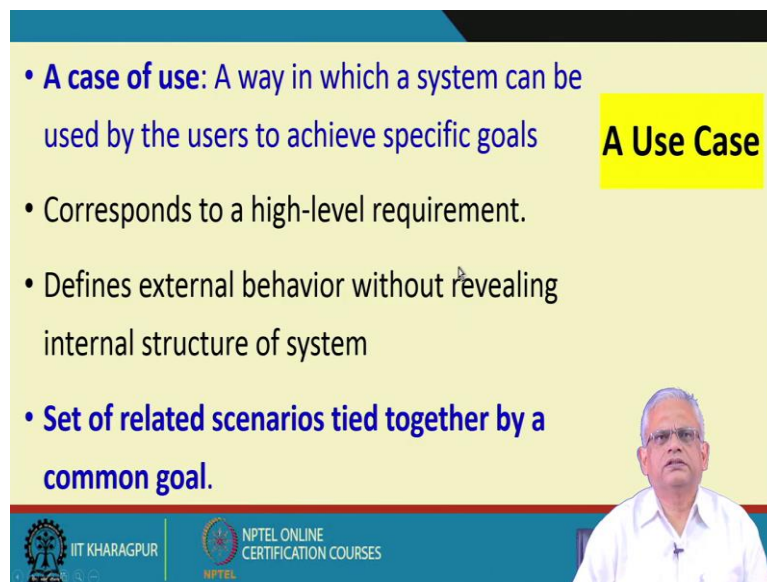
- Consists of a set of **"use cases"**
- It is the central model:
 - Other models must conform to this model
 - Not really an object-oriented model,
it is a functional model of a system

The diagram shows a central yellow circle labeled "User's View - Use Case Diagram". It is surrounded by four quadrants: "Structural View" (Class Diagram, Object Diagram), "Behavioural View" (Sequence Diagram, Collaboration Diagram, State-chart Diagram, Activity Diagram), "Implementation View" (Component Diagram), and "Environmental View" (Deployment Diagram).

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

As we discussed in the previous lecture, the Use Case Model is typically the first to be developed during system development. Based on this, the other views are constructed: The Structural View, Behavioral View, Environmental View, Implementation View, and so on. But we must understand that even though it is object-oriented modeling, but then the Use Case Model is an exception. It is actually a functional model of a system. It models what the functionalities to be supported by the system are. This is unlike the other models where we actually only model the objects and the issues associated with objects. However, in a Use Case Model we only identify the functionalities supported by the system, and we document functionalities, and therefore it's appropriate to say this is a functional model of a system, not an object model.

(Refer Slide Time: 07:37)



The slide features a yellow background with a blue header and footer. A yellow box on the right contains the title "A Use Case". The main content consists of four bullet points in blue text. At the bottom right, there is a small video inset of a man in a white shirt. The footer contains logos for IIT Kharagpur and NPTEL Online Certification Courses.

- **A case of use:** A way in which a system can be used by the users to achieve specific goals
- Corresponds to a high-level requirement.
- Defines external behavior without revealing internal structure of system
- **Set of related scenarios tied together by a common goal.**

Let's discuss some simple term. The Use Case diagram consists of many use cases. But what exactly is a Use case?

A use case is a case of use the meaning of this that if we have a system, then the use case depicts a way in which the users will use the system. Just to give an example, if we have a library software installed in our library then the different ways of using the library software will be: the members will query availability of books, they will issue books, return books, the librarian may create books, might create members, delete members etc.

So, these are different ways in which the library system is used, or these are different cases of use of the library software, and each of these (issue books, return books, etc.) is a use case. And each use case corresponds to a high-level requirement. Each use case describes a

behavior of the system that what the user can do, what input can do, and how the system will behave but then we can say that this is a black box view of the system in the sense that the system's internal structures are not revealed. However, if we look deeper into a use case, a use case consists of several scenarios.

So, let summarize this that a Use Case Model consists of a number of use cases. But if we look at each use case, we can see that each use case has a number of scenarios that tied together by a common goal. What we mean by this is that, let say you want to issue a book. So, the 'issue book' is a use case. It is a case of use by the library member. But then there can be different scenarios of issue. In one scenario, a member tries to issue a book, produce the library card, scan it, record the receipt printed, and get the book. But in another scenario of use, the member produces the book and the library card, but the system displayed that you are over quota, cannot issue the book. It may display, "Please return some books, and then only you can issue the book." So that is another scenario. A third scenario maybe, once the member tries to issue a book, and the member card, it was scanned and the system displayed that "the book is a reference book, cannot be issued out." In a fourth scenario for the same issue book use case, maybe that the member took the book and the scanner scanned the card, and then the system displayed "your membership has expired." Please renew your membership, and then you can issue the book and so on.

So, for the same issue-book use case, we can have a number of scenarios and that is very common. Every use case typically has a number of scenarios of use.

(Refer Slide Time: 12:08)

–Use cases for a Library information system

- issue-book
- query-book
- return-book
- create-member
- add-book, etc.

Example Use Cases

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, we just took an example of a library information system and here the use cases are: issue-book, query-book, return-book, create-member, add-book et cetera. These are all use cases. We need to document each of these in the Use Case Model of ‘library information system’.

(Refer Slide Time: 12:45)

- Use cases appear independent of each other
- However, Implicit dependencies may exist
- **Example:** In Library Automation System, renew-book and reserve-book are independent use cases.

–But in actual implementation of renew-book--- **A check is made to see if any book has been reserved using reserve-book.**

Are All Use Cases Independent?

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

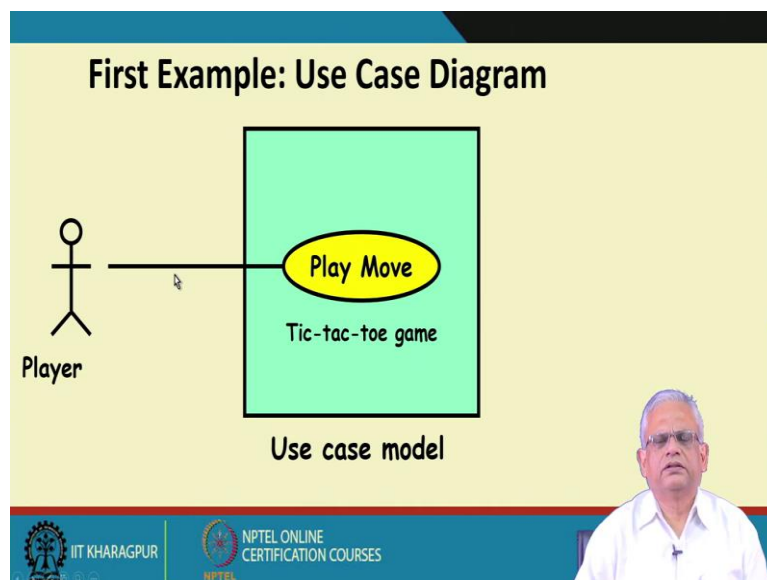
But one question that comes to mind is that if we identify use cases and these are the functions to be performed by the software. But then are these independent? Can we just have any functionality invoked by the user irrespective of the other functionalities that have already been invoked?

If the effect of one functionality depends on another functionality that was invoked, then we say that there is a dependency. When we draw the diagram, we draw them as if they are independent of each other. But we must remember that use cases are actually dependent. Some of the use cases at least are dependent on each other. To give an example, let there are two use cases in the library software: One is renew-book, and the other is reserve-book. Now let's say one user has invoked the reserve-book use case and has reserved a particular book. Then, the member who had already taken that book, once he tries to renew the book, will not renew basically because of the reservation of the book done by another member.

So, the reserve-book impacts the functionality of the renew-book and they are dependent. Even though they are represented as independent use cases, but internally, they are dependent. So, during the working of the renew-book, a check is made to see if the book has been reserved by using the reserve-book use case.

Just to summarize that when we represent the use cases, we represent them as if they are all independent. But we must understand that there may be dependencies among some of the use cases.

(Refer Slide Time: 15:25)



Now let us look at the first example of the Use case diagram (above slide). This diagram is straightforward. Even a user who is not familiar with system development should also understand the diagram and give his comments. The above slide shows the Use Case Diagram of the system 'Tic-tac-toe'. This is a software. We draw the software's boundary to indicate the functionalities or use cases supported by the software, and each use case we draw here as ellipse and write the name of the use case.

This software Tic-tac-toe game is extremely simple software and has only one function that is 'Play Move', and that's the only use case here. The user we represent in the Use Case Diagram by a stick icon symbol (left side of the diagram), and we write the user's class, that is 'Player' here. The type of the user here is player. We might have many other use cases here for configuration of the game, for sophisticated game, etc. Maybe we can have a system administrator or a program administrator who can configure the game. Then we will draw another user with a stick icon, and we will write the user name as 'system administrator' who can invoke the 'Configure Game' use case.

So, each use case is represented as an ellipse. The user is represented by stick icon. There can be more than two classes of users. These are not individual users. We can observe from the user name is that all these are collective nouns (Player, System administrator etc.), not proper nouns. These are collective nouns because they are just the type of user. Now, the line that is drawn between the use case and the user we call it as the communicates relation. For example, in the above diagram, 'player' connected with 'Play Move' use case using communicates relation. So, the diagram is very simple. The big rectangle in the Use Case diagram represents the software. Inside the rectangle, we write the use cases. The use case represented as ellipse, and the name of the use case annotated on each of the ellipse. Furthermore, from the communicates relation line, we can identify which user uses the which use case.

(Refer Slide Time: 18:22)

- Serves as requirements specification.
- How identifying actors helps in software development?

Why Develop A Use Case Diagram?

–Identifies different categories of users:

- Helps in implementing appropriate interfaces for each category of users.
- Helps in preparing appropriate documents (e.g. users' manual).

Now let us just try to answer some questions that we might have in mind.

what is the role of the Use Case Diagram?

We just saw one Use Case diagram. This is actually the starting point. The Use Case Diagrams serves as the requirement specification, a graphical requirement specification and as we proceed in this course, we will see that based on this model, the other models will get developed. We will refine this model into the other models.

But then another question is that, we can understand that the use cases themselves are elaborated, are refined into the other models, but how about the actors?

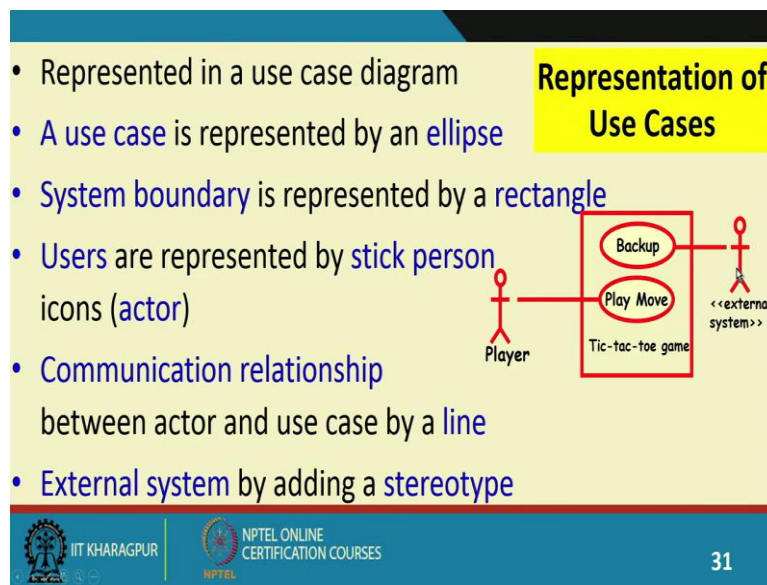
For example, we identified the player, administrator, and others as the users or actors of the software. But how does this help? Surely, we can say that identifying the specific type of user does not help in the implementation. That's appears from a first understanding. But then if we look deeper, we will understand identifying all the users in a Use Case Diagram is necessary. In the above slide, a Use Case Diagram is shown, and we can identify from the diagram various categories of users are there in the diagram. We can see from the diagram, 'Customer' user uses three use cases and 'Salesperson' uses two use cases. By identifying the different categories of users, we will know who will actually use this functionality in the later development. Is it a factory worker, or is it the manager, the technical administrator and so on. Identification of this helps because, finally, during the user interface development, we normally develop different types of interfaces for different user categories. If the user is a factory worker, then the user not conversant with the software. In this case, we will try to give a very simple interface for the factory worker. For the technical administrator, who is very familiar with the computer we will try to give an interface which may be difficult, not very so user-friendly but the administrator can execute the software very fast. He can invoke the functionality very fast even though it may not be very user-friendly.

So, for the technical administrator, the speed of use is more important than the usability issue. Another example suppose we have manager as user. We know the manager uses software almost every day and has some familiarity with software. So, for him, we will develop a different type of interface. So, identifying the users definitely helps in designing the interface. It also may be useful in developing authentication login information that who will access, which user will see what functionality on his login and also it helps in preparing appropriate documents. In case of documents preparation, identifying the user is also essential. For example, the document that is prepared for the factory worker has to be understood by the factory worker. So, it has to be in simple language with illustrations, diagrams and so on. For

the technical administrator, the manual can be more technical, and it may be something intermediate for the manager.

So, now we can understand that identifying all users in a Use Case Diagram is an important task.

(Refer Slide Time: 22:39)



Now let see how the use cases are represented. We have already seen a diagram. We have seen that use case is represented by an ellipse. The boundary of the system is a rectangle. It shows that what are the use cases contained in the system. The users are represented by a stick person icon. In the terminology of UML, we call the users as the actors. So, in the Tic-tac-toe, we have this player is an actor, and the Play Move is the use case, and the line are called as the communication relationship between the use case and the actor, and we can also model external systems that are software in Use Case Diagram.

A developed software may take the help of already existing software and hardware. For example, we might have a backup functionality that is invoked once the Play Move is complete, the backup gets invoked and the backup is taken on an external system. The external system is also represented by a stick icon.

The objective of UML was to have a simple set of notations. If we have too many notations, then it becomes very difficult to draw the diagram, to understand the diagram and so on. To simplify a language, we need to minimize the number of symbols that are supported by the system. So, we use the same symbol for an actor irrespective of the external system or

internal system. For every type of user, the same actor symbol used. So, the external system is also an actor, but we write it within '<< >>' this symbol called guillemet. In the above slide, we can see how external system is represented. So, the actor is stereotyped or we give an attribute of the actor saying that it is an external system.

(Refer Slide Time: 25:47)

What is a Connection?

- A connection is an association between an actor and a use case.
- Depicts a usage relationship
- Connection does not indicate data flow...

The diagram shows a red stick figure actor connected by a red line to a red-bordered box representing a system. Inside the box is an oval labeled 'Play Move' and the text 'Tic-tac-toe game' below it.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

But what about a connection? We said that the line is a connection. We say that the use case and the actor are associated by having this line drawn and it actually represents the usage relationship between the use case, and the actor. But one thing we must need to remember that this line does not really indicate that some data is input by the user. It may be that just invokes it, may just clicks on a button and invokes the use case, need not enter any data, just invokes.

With this very basic introduction to the Use Case Model, we will stop here, and we will continue in the next lecture where we will elaborate this Use Case Diagram with more notations, and examples and we will try to model slightly more complex software.

So, we will stop here and continue in the next lecture.

Thank you.