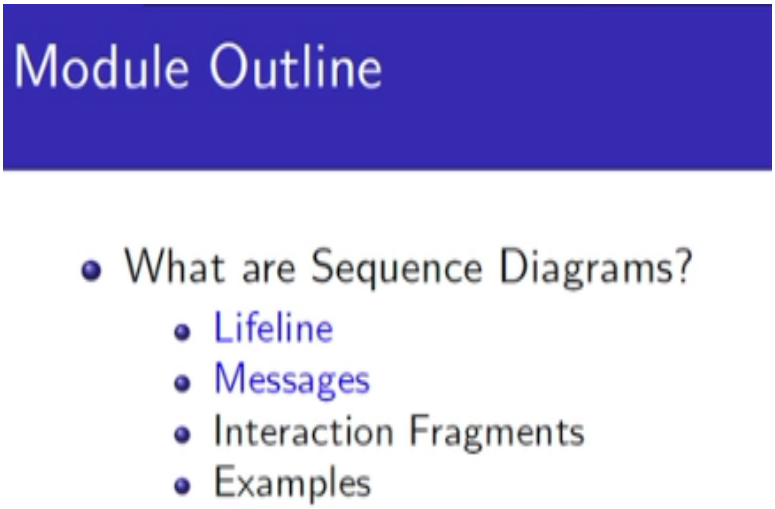
Object-Oriented Analysis and Design
Prof. Partha Pratim Das
Department of Computer Science and Engineering
Indian Institute of Technology – Kharagpur

Lecture - 41
Sequence Diagrams - Part 1

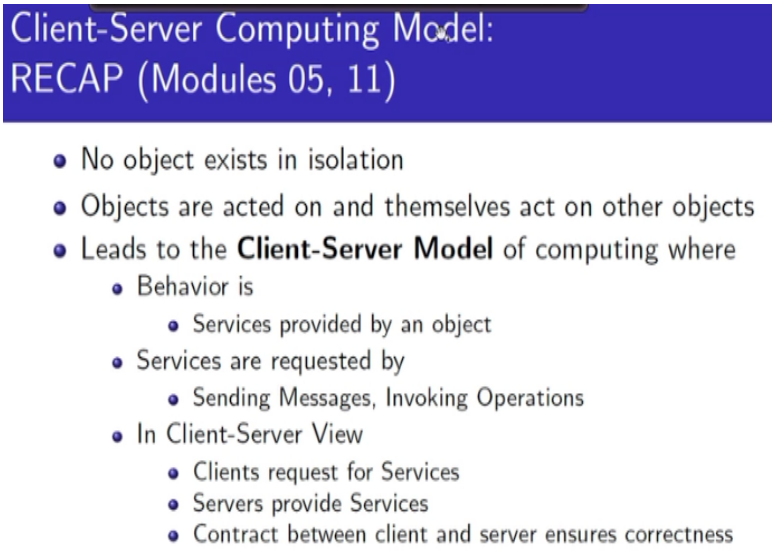
Welcome to module 29 of object-oriented analysis and design. We have been discussing about UML diagrams and we have already introduced the use case diagram, which is critical for the requirement is and we have introduced the class diagram, which is the most used structural diagram in the UML diagram set. In the present module, we will talk about sequence diagram, which is one of the behavioural diagrams, specifically categorized as an interaction diagram.

(Refer Slide Time: 01:00)



This discussion will span the current module as well as the next module, so while introducing the sequence diagram, we will talk about the lifeline and messages and discuss examples in that context and in the next module, we will talk about the interaction fragments and some more examples.

(Refer Slide Time: 01:25)

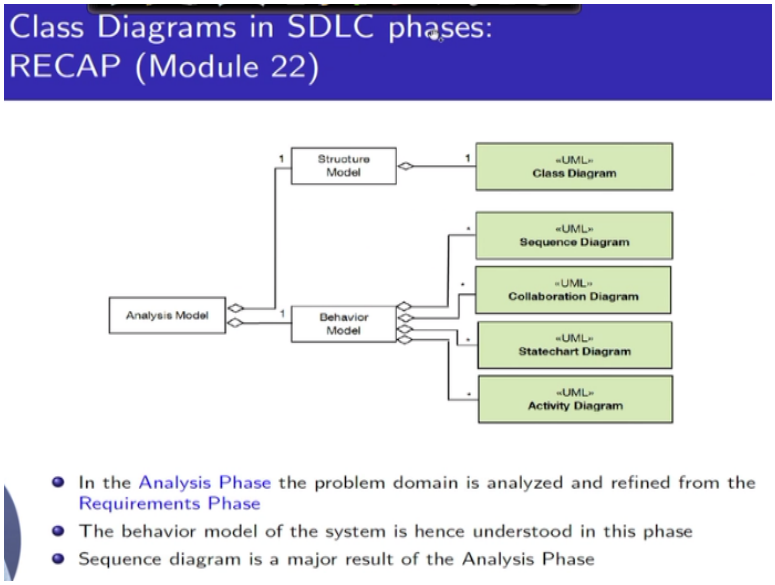


Before we get started in terms of what is a sequence diagram, I would like to remind you of the basic computing module. We have been talking about the structural aspect of it, but now we want to talk about the behaviour or execution aspect of that. So please be reminded that we have in the final module of our computation, we have several objects, so objects exist in the system. See these are different objects and this you can think of as a time.

So objects exist on that time and then, they exchange messages between themselves that one object may send some message to another object, receive a message from the object and when you get a message, you perform certain computation that is the computation associated with a particular operation, which has been requested through the message. If you recall, this was a basic client server module.

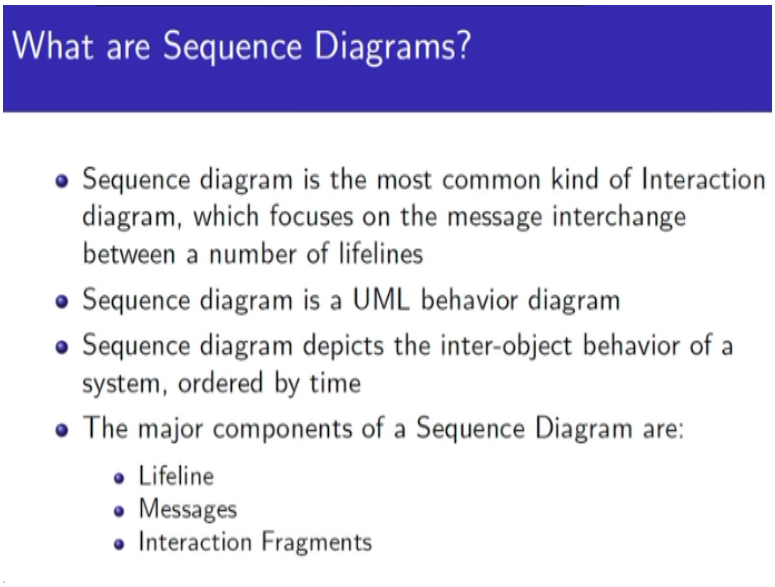
So sequence diagram is a first level by where we try to capture this behaviour of the services provided by the object and how messages request for certain service and how that service is responded. So we are going closer to realizing and representing the actual client server version of the system that we are designing.

(Refer Slide Time: 03:07)



Now, these are the different places where the sequence diagram can occur. If they can occur in the analysis phase, as we have already seen, they can occur also in the design phase as a behavioural module. So these are the two places, they start featuring from the analysis phase, where we first try to capture the dynamic behaviour of the objects.

(Refer Slide Time: 03:44)



And in the design phase, certainly we would like to refine them more. Now coming to what is a sequence diagram, as I already mentioned, it is a most common kind of interaction diagram, which focus on the message interchange between a number of lifelines. So these are the two key ideas, concepts that we will try to introduce and detail out on the sequence diagram. It is a behavioural diagram, as you know and it tries to depict.

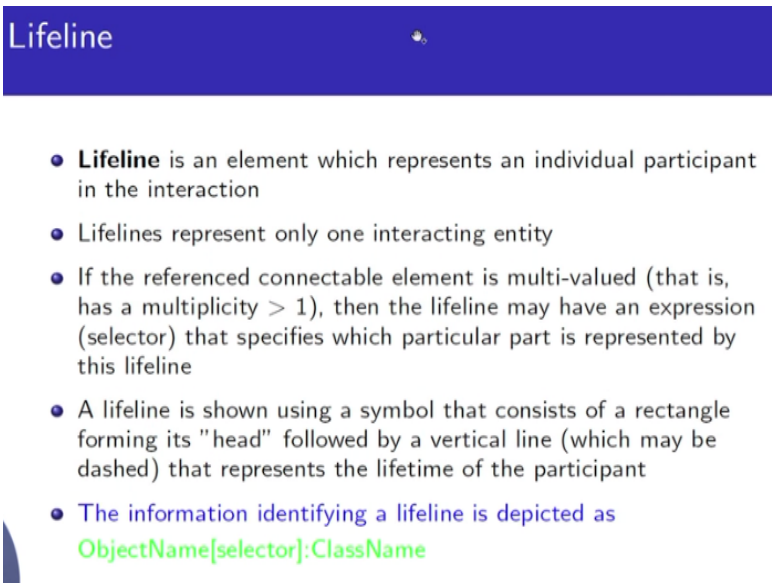
And this is the basic client server model, so it tries to depict the inter object behaviour of the system and it is ordered by time. This is very, very critical and that is why often we say this is a temporal interaction diagram. So it tells you in the order of time how things happen and when you talk about time here, we will see often that the time is not actually a clock.

That is time may not mean that so much specifics, so many seconds, so many minutes or hour has to elapse between two happenings, two events, but certainly it emphasises in the order in which things should happen. For example, if I want to check mail, then certainly I cannot do that without launching the browser. Now after launching the browser, I need to fine the URL of the Gmail. Only after I have found the URL of the Gmail, I can put in my login request.

Once I have put in the login request, it has to get authenticated by the login server. Once this is authenticated, then my inbox will be shown, so these series of things that we regularly do in terms of checking an email is a specific illustration of what a sequence diagram tries to capture. So this is the time sequence. So there is no specific time given between launching of the browser and the browser actually showing a page.

Or in terms of requesting an URL and actually getting that login page for the Gmail and so on, but the temporal ordering is most important. There are three major components of a sequence diagram, lifeline, messages, and interaction fragments.

(Refer Slide Time: 06:18)

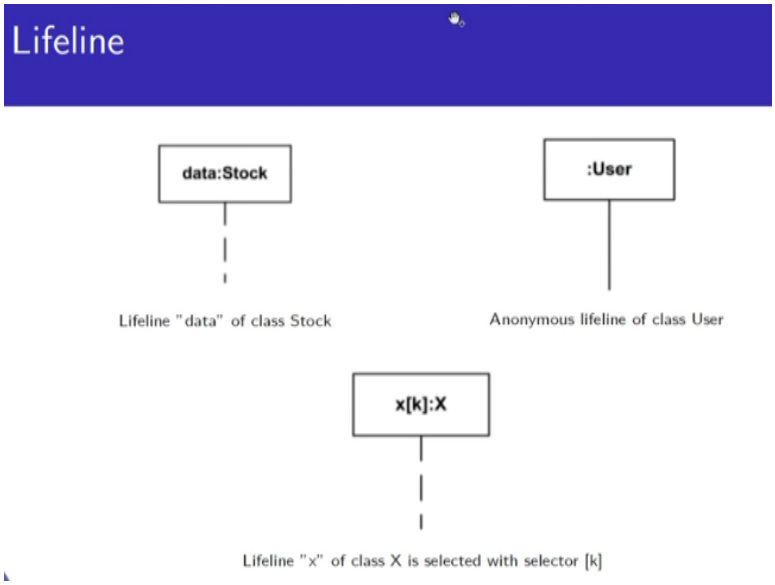


So first let us look at a lifeline. The basic concept of lifeline is every lifeline represents an object, an object in existence. So lifeline is basically the time through which the object lives

in the system. So that is generically defined as an individual participant in the interaction and it represents only one interaction entity. So if we have a collection of them, then I must use some kind of a selector to select which particular one I am talking about from the collection.

Lifeline is shown by a head, which has typically the name and then it has a dotted dashed line that goes below, the vertical line and the sense of time on that line goes from top to bottom is the advancing time. So this is where the object has started leaving and it continues till the object is annihilated. So let us start taking examples.

(Refer Slide Time: 07:28)



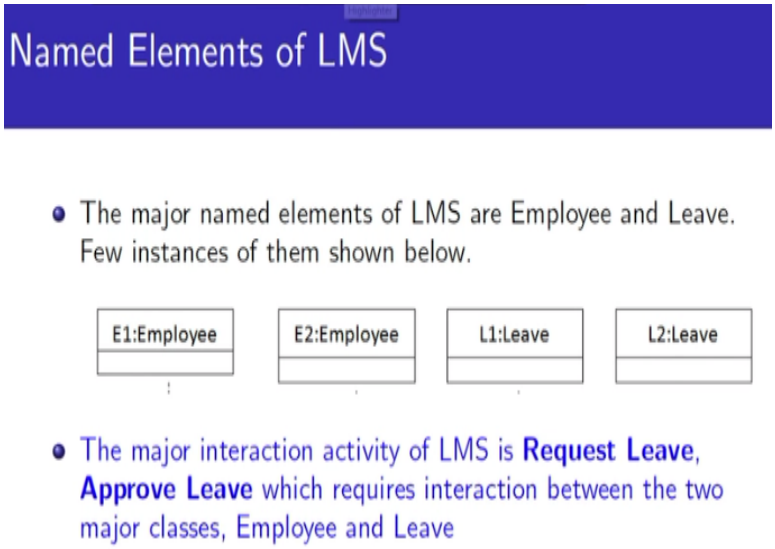
And we will be able to see. So here, this is a lifeline of data of class stock. So this notation is again important to note. These are name of a class and when I receive that with another identifier with a colon separator, then this means, this is an object. This is an instance of the class. This is the class. So data colon colon stock here in terms of the class diagram and in terms of the sequence diagram, the UML in general, would mean the data is an object of the type class.

So as I said, the lifeline will belong to an object. So this is the object and this is its lifeline, this is advancing time. The second one here is an anonymous lifeline. Here we just have the class name, but there is no object name given. So the name of the object is not specified. So this is called anonymous object or anonymous lifeline. We could write something like this as well, where certainly this is the class and this whole thing is an object.

But just note the array like notation being used here, so here possibly there are several objects of the same type and we are using a selector, this is a selector, a selector key to specify which of these objects is participating in this interaction. So these are the typical lifelines that will be involved in a sequence diagram and in all these, I should go back, these are the named entities in the UML. So you have a specific name given for classes and named entity.

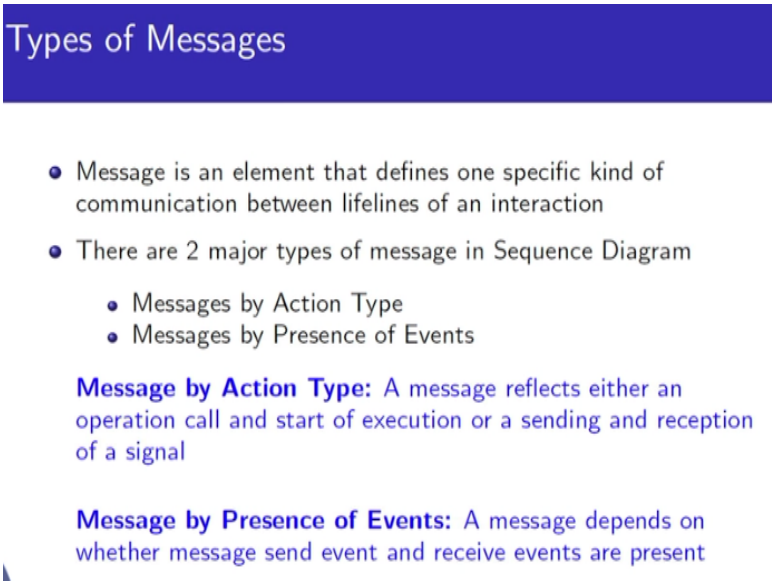
We may have names for the lifelines as well. So these are some of the named elements, entities of the LMS.

(Refer Slide Time: 09:21)



So here it says that E1 is an object of employee. E1 is a specific employee, E2 is another specific employee. L2 is a specific leave and so on and there could be the lifelines would run from these objects.

(Refer Slide Time: 09:47)

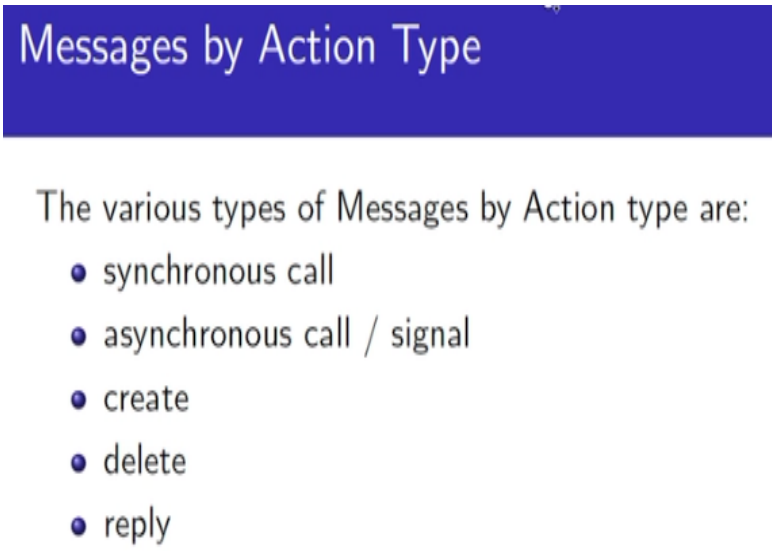


Now coming to messages, the messages and element that defines one specific kind of communication between lifelines of an interaction. So usually a message will be between two lifelines, most cases and they are categorized according to two broad classes, two major types, one is by the action type, that is what kind of role the message and messages to achieve and other is by the presence of the events.

That the message certainly is something that an object sends to another object in the generic form of the client server model. So a message usually will need to have a send event, which initiates sending of the message and will lead to have a receive event, which is the receiving of the message, so if we classify messages according to the presence of absence of these events, then the classification is called the messages by presence of events.

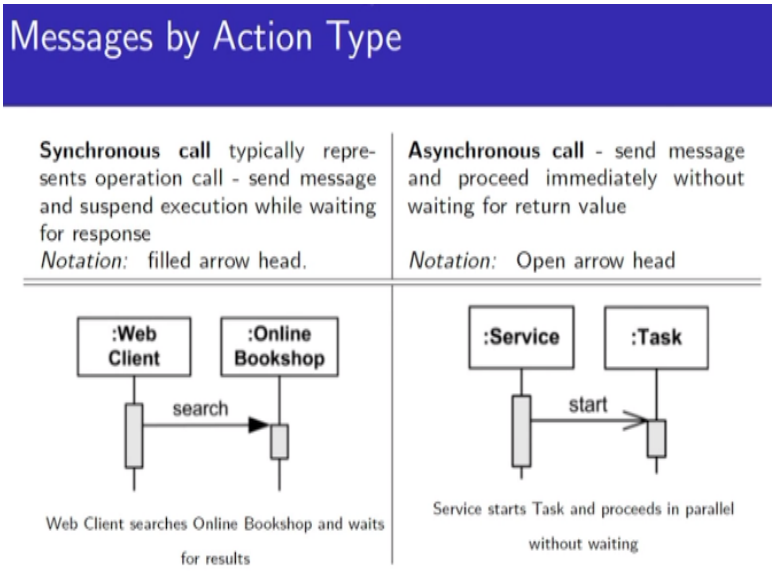
So we will take a look into each one of these types of messages, what are the different varieties that exist in every case.

(Refer Slide Time: 11:03)



So we go by action type, then various types of messages under the action type are these: synchronous call, asynchronous call or signal, create message, delete message, reply a message, so these are the kinds of messages that are allowed in a sequence diagram.

(Refer Slide Time: 11:21)



So these are synchronous call, so let me just explain the basic concept. So there is an object here, there is another interacting object here and this means something is in execution, some method of an object is in execution. So you send a message, a search message to this particular interacting object. And as you send that, the search method or the search operation in the online book shop object gets activated. So that starts executing.

You do a similar thing here as well. Here there is a service object, there is a task object, you can note that in all these cases, the objects are basically anonymous and you are executing

some method of service. You send a message to task, which is a start message, so basically the start operation in task will start executing. That is the basic structure of sending a message.

Now you do note that there is a difference in the arrow head, which gives the difference in terms of their basic behaviour. We say this is if the arrow head is filled up, we say this is synchronous call, which means that once the request has been sent, the requesting object, that is this particular method, which was executing will be put on hold till this requested operation is completed and that response comes back, some way to indicate that this has completed.

So between the requestor and the requested, only one will work at a time. When the requestor has sent the request, then this is executing, this is on hold at this point and only when this has completed, this will start executing again, that is synchronous. Asynchronous is where the requestor sends a request, this is initiated, this continues, but requestor also keeps on continuing, requestor does not stop. So requestor has sent the task, just do that task.

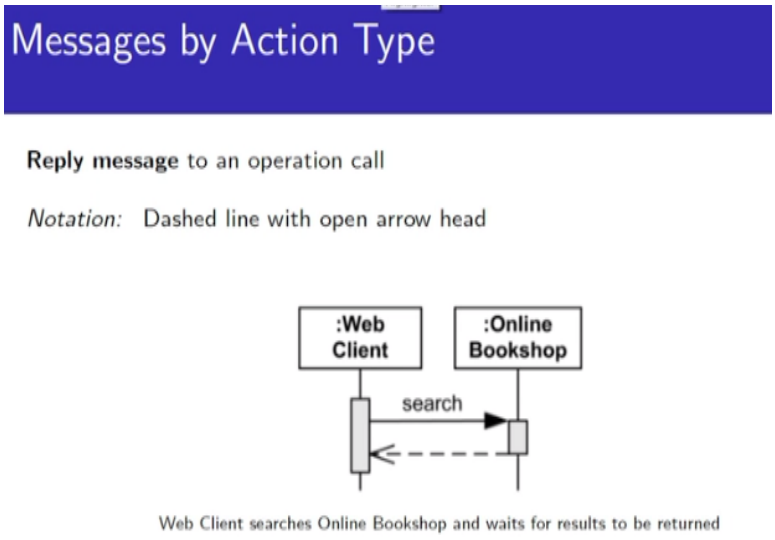
I tell somebody, this is the letter, please go to the post office and deposit that letter, send that letter, but after sending that person, I continue with my own work, so that is an asynchronous call. It is not the execution of one here is not dependent on the other. Here the execution is dependent because till it completes the execution, this cannot start further, so this is one of the key ideas, key properties of a message that you must identify when you are modelling.

As to you want to deal it synchronised or need it asynchronous. Of the other types are various, you know, certainly the basic hygiene kind of messages as I said, is a create message, so you will need to create objects at different points of time, so this is designated by a dashed arrow head and the head of it is not an execution box, but is an object itself. So this says online book shop is creating an account. So this is called a create message.

Similarly, this account exists, an online book store is sending a message, you can see the name of the message is destroyed, which does invoke, I mean if you know C++, Java, it invokes the destructor and the object is eventually destroyed. So beyond this there is no lifetime for this object, nothing exists after this. This object was created here is terminated at this point, so this is known as delete message.

So that is the basic object creation and destruction that will occur. So we have seen synchronous, asynchronous, create, delete, the only other message that is important is a reply or response message, which may be sent.

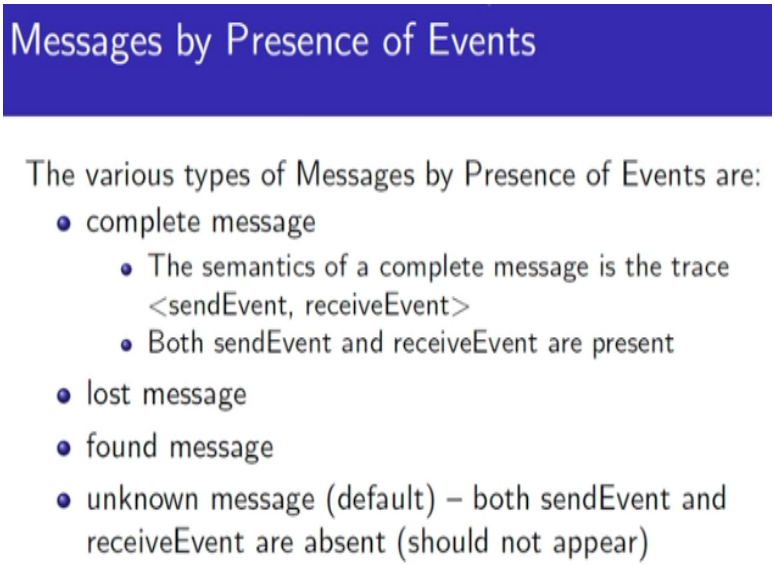
(Refer Slide Time: 15:23)



In many cases, it is optional, it is typically shown with dotted line and with open arrow, so a web client sent search request to the online book shop and online book shop in return has sent a reply. So this is a reply message or is often called a response message also. Now, it is not that always there will be response, certainly for example, for a synchronous call, the reply message may not be there because the requestor is already executing in some other state.

But in a synchronous call, he will need a response message because you remember that the requestor who is the sender of the message is waiting for the requested task to be completed and therefore that completion will have to be reported in terms of a reply message. So these are the major types of messages in terms of the actions that they perform.

(Refer Slide Time: 16:25)

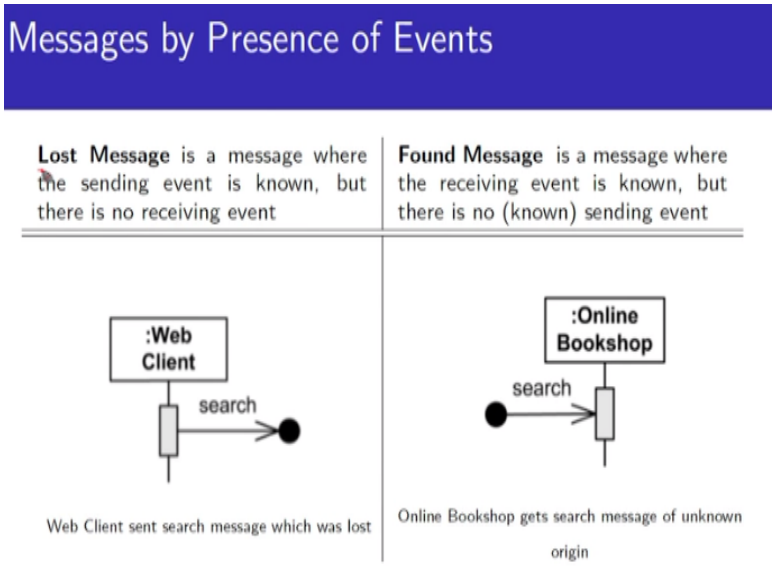


In terms of the events as I said that most messages are complete message that they have a send event and a receive event. A request has been made and by reply this has been completed in terms of that. So when I send a message, this is the point where I am doing the send, so I have a send event like this and when the message is received by the receiver, I have a receive event at this.

So normally a complete message would mean that one that has been sent by the send event and that has been received by the receiver. So because you would expect that why you are doing messages because you want to interact. So in the interaction, there will have to be sender, then there will have to be a receiver that is to be received. So if that is satisfied, then we say we have a complete message.

But there could be lost messages or found messages. As we see, the lost message is one where you just have a sender, you do not have a receiver or a found you just have the receiver, you do not have a sender and you could have unknown messages where both the send and receive events are not present. So that is the kind of a default, but certainly in a sequence diagram, unknown messages usually would not mean anything and you should not have them.

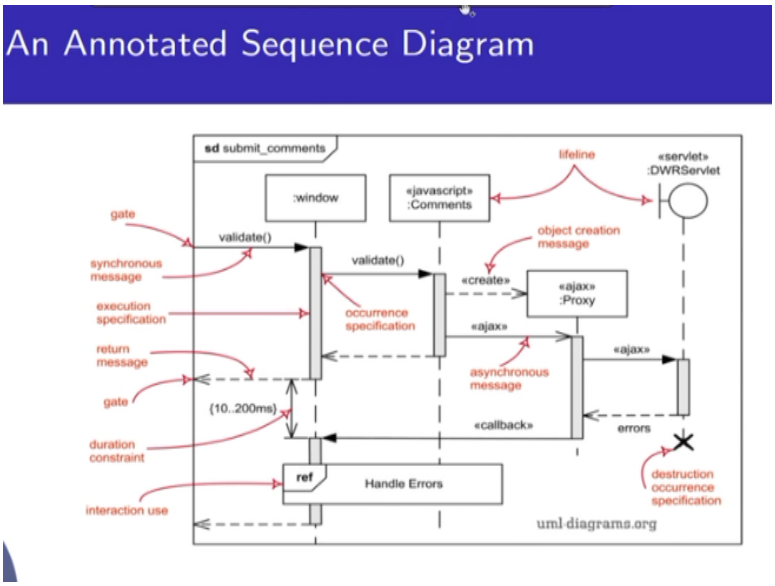
(Refer Slide Time: 18:10)



So in terms of the special kind of messages, the lost and found, lost message is one the request was sent, so here the send event happened, but it never reached anybody, which is very typical when we go to web search and all that, we have sent the request and it does not reach the web browser, it gets lost, dropped in the whole process and we denote that in terms of this big black dot. So this says that the received event was not there.

On the other hand, it could be found messages that the online book store gets a search request, so this is the received event, but there is no identified sender on the other side. It did not know where did it come from. So the send event is not known. So you say in comparison to lost, this is a found message. So these are possible message in terms of presence of events, but certainly we would primarily work with complete messages, which have both the sent as well as receive events.

(Refer Slide Time: 19:19)



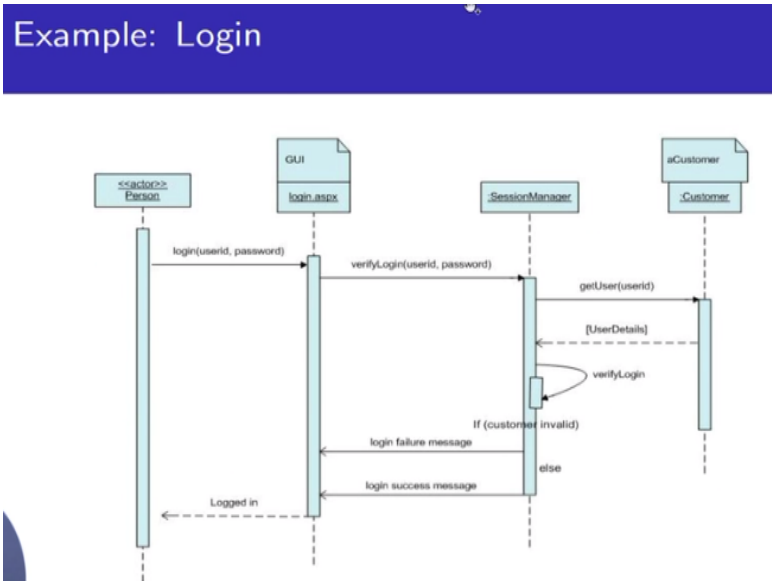
So this is just for you to see and get familiar with different messages, so we can see here. There is a lot of red in this diagram, then we will use blue. So you can see here. We can say that this window object is sending a validate message to the comments object and what kind of message is this? This is on one site, synchronous call message. What kind of by presence of event? This is a complete message. You have both the sent as well as receive.

So we can see different such types, for example here creates, so you have a create message, which is creating the proxy, the comments is creating the proxy. So you have a create message, the new lifeline starts. There is an existing lifeline of servlet and a message has been sent to do some tasks, which felt and because of that the object is destructed. So this is a delete message that you have in place.

You have a reply message here. There are several other notations. Please do not get worried about them. We will slowly introduce those. And if you particularly look at this message being sent from comments to proxy, after this was a create and this ajax message is being sent from the comments to proxy. You can see this is again a call message, but this is an asynchronous call.

So after sending this, the comments will continue to work and this execution starts. So these are the different kinds of messages, just illustration of the notation that you would think of.

(Refer Slide Time: 21:19)



So you could go through some examples quickly. For example, I want to do a login, how do I model that. Certainly, here is the initiating, here is the person who is initiating who wants to do the login, so the corresponding object sends a login request and within parenthesis what we write are known as message parameters, message could have parameters that it carry, certainly if you have to login, you need to specify the user ID and the password.

And the message is sent to the GUI because that is where we write down the user ID, password. So the message is sent to this. The message is synchronous because unless the login is completed, I cannot do anything else. Next, the GUI what it will do, it will need to discuss or consult the session manager of the particular session as to check if you are already logged in, if you are in the session, if your login is valid.

So in turn it sends a verify login message to the session manager, this again is synchronous, which means that when this verify login message is received at this point, at this event, then this as well as this, both of them are on hold in execution. They are not executed. Now the session manager has to check that, so the session manager looks at that customer database to check if the user exists, so it does it, sends it, get user details, and sends the user ID again in a synchronous manner till it gets a response of the user details.

So in this path of execution in the customer object or which is basically a customer store, so this is customer class and a customer say as if the name of the particular customer object. So the customers based on the user ID, the user details are returned. So once the session manager

gets the user details, so it knows the password that has been given, that has come from this message. It has got the user details.

So it has got the password of the user as present in the system. So at this point, it has the password from this site, it has a password from the system. Now it has to verify that whether they are same, so that is verifying the login again, but this message interestingly you can see is a self message that is the session manager actually sends a message to itself and this smaller one denotes the verify login as a self message.

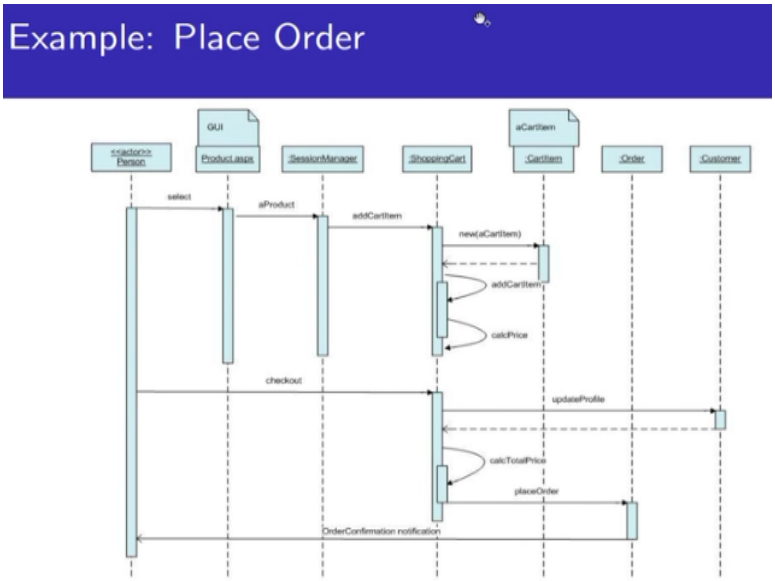
Because it is important to verify that because password is normally not kept in the way they are typed. They are kept in some encrypted form, so the password that you have got that needs to be encrypted and then it is to be compared with this encryption and so on, so that is the verify login. So you can see that the session manager on one that got the verify message from the login GUI and the other its gets a verify login again from itself and it computes and all of these are asynchronous.

Because the whole process has to go in step by step, one step after the other. And after this has been done, at this point what will you have? You will have either the login is successful, the password has matched or it did not match, so you have a branching. So this is how it is typically you will write branching. There are other ways of noting that, so you say if customer invalid, then you will do a response message, which is login failure message.

That say that the login did not go through else you sent a login success message, so you can see there are two messages that go back from the session manager to the GUI, both of these are of reply message kind. Because they are responding back, because they are replying. But only one of them can be sent depending on what happen in the execution, is it valid login or is it invalid.

Whichever comes across, then based on that the logged in information is given back to the person, you either say that you are logged in or you will say that your login has failed. So this is just a simple sequence diagram to show that the typical way we login, how that can be represented in terms of the different lifelines and messages that pass between them.

(Refer Slide Time: 26:07)



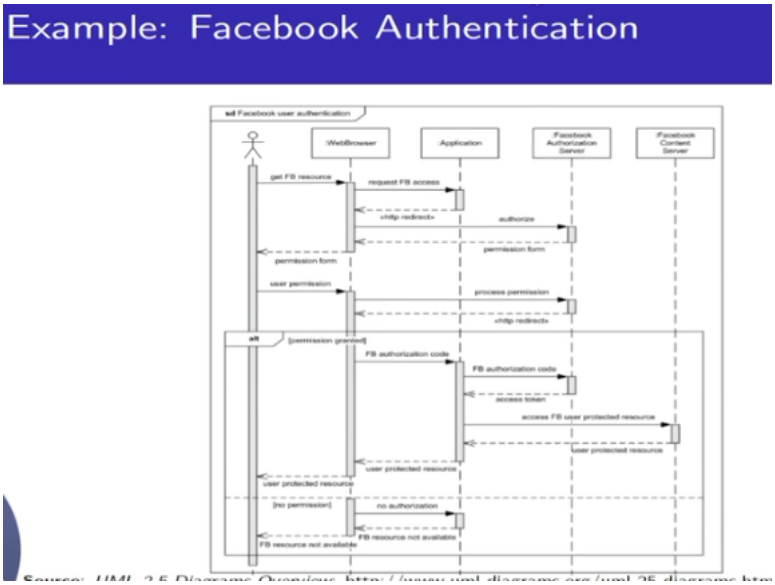
I have another illustration here, which I leave for your own analysis and understanding. It is on the similar line. So if the person has already logged in and as if this is basically some kind of an online store, so after login, the person wants to place an order, so the person is here, the GUI where the products can be seen, the session manager is there, the shopping cart where you elect the items that we have selected is there, specific cart items are there.

Then we have the order in which once, we have done the collection in the shopping card, we would like to actually place the order and the customer, so these are the different objects, which participate in this and these are the different messages that go through them. Please go through this diagram, we can see there are several of them are actually synchronous calls.

All of these are synchronous calls because in the shopping process, online purchase process of any kind, certainly things will have to go step by step and only one step is done completed, the next step can happen, so that is a synchronous situation broadly. There are several like cart item or calculate price, which are self message, which you can use and then finally you have the order confirmation notification that comes back to the person.

So I would request that you carefully go through each and every message and try to map it to your own experience of having purchased anything from the online store and you will find that how we are representing those behavioural aspects of the process in terms of the sequence diagram.

(Refer Slide Time: 27:59)



There is yet another example. This is as if you are doing a Facebook authentication, so this is the web browser through which you are working this application. This is Facebook authentication server and this is the Facebook content server. So the idea is working from the browser, you have to first approach the Facebook application, then the Facebook application whether you can access that depends on what the authentication server will tell us.

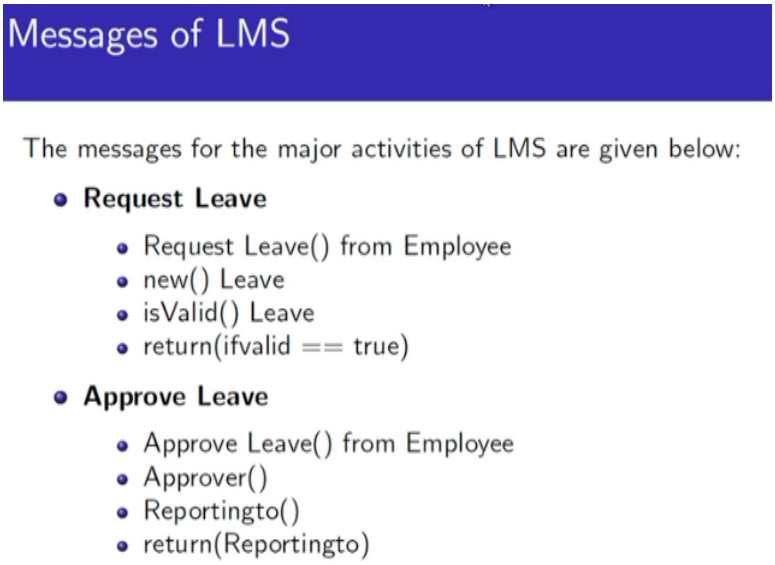
And once you are authenticated, then you can actually access the content from the Facebook content server. So these are the four major objects or four major timelines that participate in accessing an element in the Facebook and if you again look carefully at this part, this is mostly synchronised message and response, which consults the authentication server through the application to validate if you have permission for access.

And once you have permission for access, then you do this part, which is actually accessing the content, so your requests are going till up to the content server. Again you read through these sequence of messages and try to understand how this is happening and try to understand in depth. Because if you understand a couple of these sequence diagrams, in terms of your personal experience.

Because I am trying to pick up examples, which I believe that most of you who are attending this course would have personally experienced either purchasing something online or accessing certain message, certain photos, certain video on the Facebook, so these are the typical kinds of sequences that we will need to go through and this is how they can be modelled.

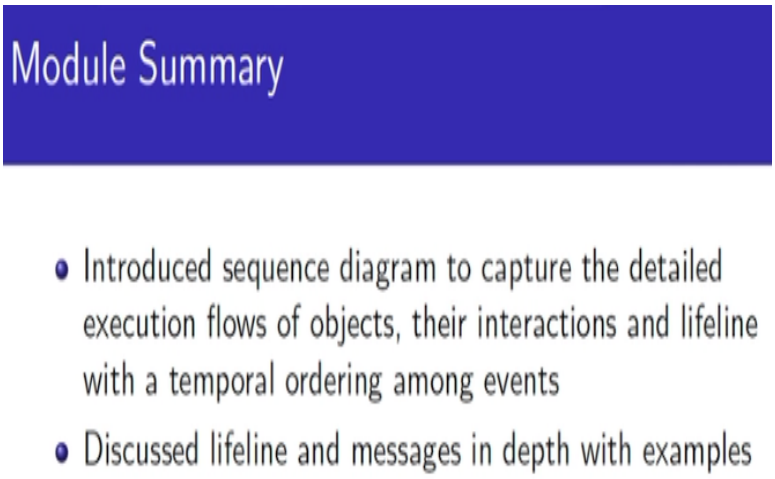
There is of course a final part where permission is not there, so you do not actually in this part below the dotted line here, you do not actually able to go up to the content server and access the content because your permission has been denied.

(Refer Slide Time: 30:03)



So that is mostly about the lifeline and messages and in view of this, if you look into the LMS, then our typical message would be a request to leave message or approved leave message, so it will come from an employee to the leave and then we will have to see what should be the different processing that should happen. So I leave it as an exercise to you to think over as to what should be the basic sequence diagram structure for a request leave or approved leave that we have discussed for the LMS system.

(Refer Slide Time: 30:39)



In summary, we have introduced sequence diagram to capture the basic interaction of different objects and their behaviour under the client server model. We have explained the modelling of objects lifetime in terms of lifeline components and discussed about the various messages, different kinds of synchronous and asynchronous, these are two key kinds of messages and the response.

These are the three key types of messages, which will need most of the time to deal with any situation of dynamic behaviour besides the create and delete messages. So I will ask that you please try to go through the example diagrams that we have shown and try to become confident about understanding the lifeline messages before you take up this next module, where we will talk about the interaction fragments and further on the LMS example.