

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

Lecture – 20
Relationships among objects

(Refer Slide Time: 00:29)

Module 12: Object Oriented Analysis & Design
Relationships among Objects

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick
Srijoni Majumdar
Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author

NPTEL MOOCs Object Oriented Analysis and Design

Partha Pratim Das

1

Welcome to module 12 of object-oriented analysis and design. In the last module we have discussed about the nature of objects. We have noted that an object is typically characterized by its state, its behavior and its identity. So with that we are now ready that in our system we will have identifiable objects as many of different object as we need to and each object will have its states which can be in terms of static properties and dynamic values.

And each object would manifest the different behavior for providing services to other object and they will interact to build up the system.


(Refer Slide Time: 01:39)

Module Objectives

- Understanding Relationships among Objects


In the current module we will talk specifically about relationship between objects. So our objective in this module understands the relationship amongst objects.

(Refer Slide Time: 01:43)

	Module Outline
Module 12 Partha Pratim Das Objectives & Outline Relationships / Associations Links Message Flow Roles Visibility Synchronization Aggregation Examples Temperature Controller LMS Summary	<ul style="list-style-type: none">• Relationship / Associations among objects• Links<ul style="list-style-type: none">• Message Flow• Roles• Visibility• Synchronization• Aggregation• Examples<ul style="list-style-type: none">• Temperature Controller• LMS

This is the module outline as usual this will be visible on the left hand side of every slide so we will proceed according to that.

(Refer Slide Time: 01:54)



Relationships among Objects

Objects contribute to the behavior of a system by collaborating with one another

Module 12

Partha Pratim Das

Objectives & Outline

Relationships / Associations

Links

Message Flow

Roles

Visibility

Synchronization

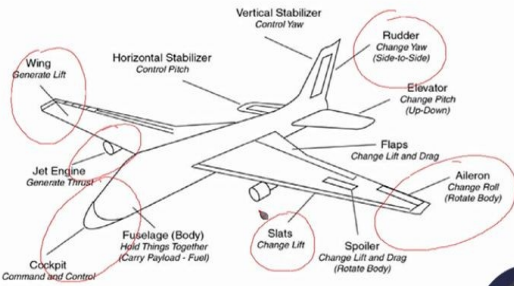
Aggregation

Examples


Temperature Controller

LMS

Summary



All the components of an airplane have an inherent tendency to fall to their collaborative efforts enable it to fly and stave off any outcome of



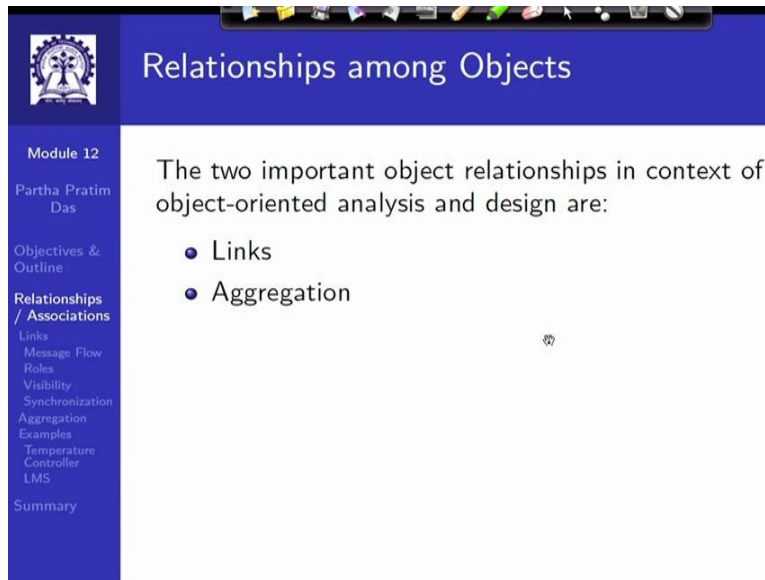
NPTEL MOOCs Object Oriented Analysis and Design
Partha Pratim Das

In terms of relationships of objects, the point that we note that in the whole system we will have several different objects that would exist. But the object one or more objects in isolation does not provide any interesting system. This is not an interesting situation to analysis or design. But when the objects come together they get interrelated they get associated link in different ways then they start showing really interesting properties.

They start giving capabilities which are critical for the total complex system that we are designing. Say for an example, here I just show an aircraft. This is as you can see this is the whole of the aircraft and it has a different component object say the cockpit, the engine, the wings, two sides, the rudder, and the slates and so on. The several different objects exist in that airplane.

Now if we look into this object individually they themselves have some complex behavior but with each by themselves do not really achieve anything till we put them all together connect them all together into a single airplane which can fly and carry people. So the relationship primarily has to talk about how the objects can get interconnected and behave between themselves.

(Refer Slide Time: 03:47)



The screenshot shows a presentation slide with a blue header and a blue sidebar. The header contains the title 'Relationships among Objects' and a small logo. The sidebar lists the contents of Module 12, with 'Relationships / Associations' highlighted. The main content area is white and contains text and a bulleted list.

Relationships among Objects

Module 12
Partha Pratim Das

Objectives & Outline

Relationships / Associations

- Links
- Message Flow
- Roles
- Visibility
- Synchronization
- Aggregation
- Examples
- Temperature Controller
- LMS

Summary

The two important object relationships in context of object-oriented analysis and design are:

- Links
- Aggregation

In terms of object-oriented paradigm, we talk of two important relationships for objects one that is links and often they are also referred to as association and the other is aggregation.

(Refer Slide Time: 04:08)

A link is a physical or conceptual connection between objects

- An object collaborates with other objects through its links to these objects
- A Link denotes the specific association with the help of *passing messages* through which
 - One object uses/invokes/applies the services of another object
 - One object navigates to another object

So first let us talk about links. The link is a physical or conceptual connection between objects. So I may have two objects which are physically connected like I have just talked about the airplane. The cockpit, the body, and the engine of that aircraft are physically connected so they are linked or there could be conceptual connection between two objects. I am an individual so I may be described in terms of an object which has a name, date of birth, designation, affiliation and so on.

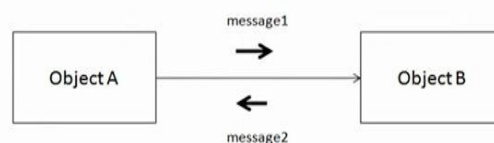
I have a bank account with some bank where my account has the account number the type of account, the transactions, the account balance and so on. So there is a relationship between me and my account this is also a kind of link but this link is a kind of conceptual image. There is no physical link between me and my account but certainly I am related to my account. My account related to me. So that is a basic notation of links.

An object collaborates with other objects so it says this in formal terms an object collaborates with other object through its links and this links if we refer back recall a basic model of object-oriented systems. The model is the client-server model where two objects one needs a service and the other which will provide the service. The one which we call the client which needs the service and the provider the server object will pass messages between themselves to achieve what is required.

So links necessarily show that where the message can pass from one object to the other. It could be a use kind of a message invocation kind of a message or application for services kind of a message or it could be just navigation from one object to the other. So link basically is a representation of the interrelationships between two or more objects.

(Refer Slide Time: 06:34)

- A link is depicted by a line between two objects
- The direction of the line, depicts which objects, invokes the services of the other object
 - If there is no direction, then the link is bidirectional, both the objects invokes each other's services
- A message can be unidirectional or bidirectional in a link
- A message is represented with a small directed line and label to define the message



Object A invokes the services of Object B



In this links are more often depicted in some graphical notation and we will start introducing this and slowly we will see at a later point that refinement of this notation will result in to our unified

modeling language diagrams. The UML diagrams. So here we are just starting with every simple notation where I have one object which is just written shown as a rectangular box. There is another object and there is a link connecting them.

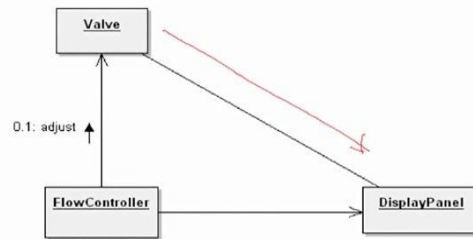
So this says that there is a link there is an association between object A and object B and in this case please note that there is a direction given so it is the link is direct one. It says that object A can ask object B to provide a service. So the object A can pass a message or invoke services that object B provides. So that is done in terms of sending the request message from object A to object B.

And so there is two kinds of directionality that is involved here one is the direct of the link which shows who is requesting this object is requesting and this object is responding. And then on top of that we are showing the direction in which the message goes certainly the client object who is requesting has to send a message to the server object so that direction is same as the direction of the link. Now in some cases the receiver object or the server object will need to send back some result or some response.

So there could optionally be a message which goes in the reverse direction of the link as well. So on one side the link shows who is the client? And who is the server? Who is requesting and who is serving? The message direction on that could be in the direction of the link that must be there because that is how the request is made. But there could also be bidirectional flow of messages over a particular link.

(Refer Slide Time: 09:00)

A Flow Control System



Links in a Flow Control System

Source: *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007)

- The flow control system has three objects primarily: FlowController, Valve and DisplayPanel
- The FlowController object has a link to a Valve object
- The FlowController object has a link to the DisplayPanel object
- The Valve object has a link to the DisplayPanel object

NPTEL MOOCs Object Oriented Analysis and Design

Partha Pratim Das



So with this let us look at a very simple example so this is we are talking about flow control systems has three objects. There is a valve. So it is basically the application is like this possibly there is a pipe through which water is flowing and that flow needs to be control that you can increase the amount of flow. You can decrease the amount of flow and so on. So the object through which you do that is called a valve.

And certainly there is an object which we model here as flow controller who decides as to how much flow will happen so that controls the flow so it can adjust the flow that is happening and the third we are also talking about a kind of object say like display panel where we can see what is the current flow that is happening. That is kind of a simple model so you can think of I am specifying this in terms of a flow control system.

The similar thing you find in terms of the electrical metre that we have at home and so on. So if we look in to the specifics of this so it has three objects as I have already mentioned. The flow control object has a link to the valve object. I am talking about this particular link and the direction of this link say that the flow control object can sent messages to the valve object. So this will be messages like adjust.

So the flow control will say that you adjust you increase the volume decrease the volume. You can increase the volume by certain number of units and so on. So the link from the flow control

object to the valve object shows that this can work as a client and ask while this works as a server ask it to adjust the flow. In a different way there is a link from the flow control object to the display panel object.

When the flow control object has adjusted the flow the flow control object would send the message to the display panel say that to display that this is going to be the flow that is going to happen. Or it could also be that the flow control sends message to the valve saying that this is the adjustment and then valve as it gets adjusted to the new flow sends a message to the display panel and shows that message.

So that this link shows that there could be a message from the valve to the display panel as well. So this is the basic way to show how different objects can be linked between themselves.

(Refer Slide Time: 11:56)

Understanding Messages between Objects

Message Exchange Between **FlowController** and **Valve**:

- **FlowController** invokes operations on the **Valve** object, but the flow of message can be bidirectional like request and response messages
- When **FlowController** invokes the operation *adjust* (message *Adjust* sent) on the **Valve** object to update the settings, then data flows from **FlowController** to **Valve**
- When **FlowController** invokes the operation, *isClosed* on the **Valve** object, to check if the **Valve** is closed, then the **Valve** object sends the response, hence data flows from **Valve** to **FlowController**

So if we look in to the messages that exchange the flow controller invokes the operation on the valve object here again please note that the message can be bidirectional because it is requesting the valve to do something. So say if it is adjusted then possibly the flow controller will simply say okay adjust increase it by two units, decrease it by three units something like that. But suppose it wants to send a different message to the valve object saying it is closed.

So it is trying to check if the valve at all is working or it has been closed down. So naturally the flow controller need now in this case expects a response back from the valve object as to what is the state of the valve object is it flowing or is it closed it is stopped. So even though there is a single direction which is sending the message is closed it will expect a message that which will say may be say true or false something like this.

So the message flow in this case could be bidirectional. On a single link of one direction there could be multiple directions of messages that can fly between them.

(Refer Slide Time: 13:16)

Roles of Participating Objects in a Link

- The **FlowController** object operates on the **Valve** object and **DisplayPanel** Object (to change settings or display string respectively); whereas the **Valve** and **DisplayPanel** objects do not operate on the **FlowController** object. Hence it is known as the **Controller** or **Active Object**
- **Separation of concerns** is achieved by the **FlowController** object
- The **DisplayPanel** object, does not operate on other objects; it is only operated on by other objects. Hence it is known as the **Server**
- The **Valve** object, can operate on **DisplayPanel** and get operated by **FlowController**. Hence it is known as the **Proxy**

Now we also characterize so this is we just took a look at how the messages can flow. How that will depict in terms of connecting the object through links now from this link diagram of message we can talk about what is a rule that this participating object play in terms of a link. So some objects like the flow controller basically sends out messages to others to get some service whereas it by itself the flow controller does not serve anyone so such objects are known as controller objects or active objects.

They kind of control the system; they kind of drive the system. We will see more specifically as to what is the definition of active objects what characterizes flow control in this case is a fact that other objects do not operate on it. But the flow controller operates on the valve and the display

panel. The display panel on the other hand does not operate on any other object. It does not ask for a service from anyone else.

It will just provide service to others that is whether the flow controller asks it to display some information whether the valve asks it to display the flow rate it will comply and display that. So such objects which only provide service are known as the server objects and if you look at the valve object, the valve object actually does kind of both. It provides service to the flow control because it can receive message from the flow control to adjust the flow to respond to flow control as to whether it is closed or open.

At the same time, it could send request to the display panel to display the state of the valve to display the amount of flow that is happening in the valve. So these are kind of object which at the same time can operate on some other object and can be operated by other objects such objects are known as proxy objects. So we have seen that we have primarily flow controller or active object flow who only asks for services.

These are commonly called clients. We have server objects who would predominantly provide services to others without asking from service from others. These are known as server objects and the proxy objects are kind of mixture of them. The identification rule we will see later on would help us in actually finding out how this different object should be implemented.

(Refer Slide Time: 16:25)

Visibility

If the **FlowController** invokes operations on the **Valve** object, the **Valve** object should be visible to the **FlowController** object

We need to check the visibility across links because the scope and access of the objects on each side of a link are dependent

Types of visibility are:

- public
- package
- protected
- private



The next parameter that we need to know is that the objects also has visibility factors. For example, the flow controller has to send a message request to the object valve. Naturally that means that the flow controller must be able to see the valve object. Flow controller must know where the valve object exists and it should be able to send that message. Visibility basically means that whether it can actually send the message whether it can actually send the request.

So it is important that between the objects the links alone will just say whether there is a possibility of sending a message but will also need to make sure that in the whole pool of object that exist in a system there is proper visibility between them. And there are four types of visibility known by public visibility, package visibility, protected visibility and private visibility. We will not elaborate on this a lot here.

But we will come back to this in a later module after we have discussed about the classes of different modules.

(Refer Slide Time: 17:56)

Synchronization



Whenever one object passes a message to another across a link, the two objects are said to be synchronized – they are *Active* and *Passive* Objects.

Object Type Illustrative Behavior

- | | |
|----------------|--|
| Active | <ul style="list-style-type: none">• An object which instigates an interaction in a thread• It can change its state by itself and send a message to other object if necessary• It is responsible for handling control to other objects• It is referred to as a Client |
| Passive | <ul style="list-style-type: none">• An objects which passively waits for the message to be processed• It is activated when it receives a message from other object• It waits for another object that requires its services• It is referred to as a Server |



The next concept in connection to relationship amongst the object is the concept of synchronization. And this is a concept which you need to understand every carefully because this is little different from the typical programming concept where we know that we have written a code and we start executing it if you have written it C or Java kind of language then we know that there is main function from where the execution starts.

Different functions are called and when the task is over this particular main function terminates. But in terms of an object-oriented system when that comes in to be please remember that we have multiple objects. So if we try to kind of look at the model we have multiple objects I have drawn similar figures very often. We have multiple objects and the objects are sending messages they are sending post back response and so on.

So what we need is when object one who is the client is sending a message to an object O2 then both object O1 and O2 must be in execution. That is unlike the typical programs that you may have written and executed earlier which are called sequential programs or single threaded programs. Here we need different executions happening associated with different objects. Because in the time when O1 executes O1 is doing its operations at the same time O2 will have to do its operations as well.

So these are typically called distributed systems or concurrent systems to be more precise. And whenever you have that there is a need for synchronization because if two threads or more threads are trying to work on the same objects or the same set of objects because there are different objects running on them then there is always a question of issue of whether the operation will be correct whether the operation is performed in the right way or not.

So we will soon discuss an example to illustrate this further but before that let me introduce two different types of objects in the context of synchronization. They are called active objects and passive objects. An active object is this is the definition and an active object is which instigates an interaction in a thread. That is what it is simply means that an active object by itself will decide that something needs to be done and we will start doing that.

So the thread on which the active object runs will by itself manage change its state and for doing whatever it is to do send message to other objects. Naturally they are responsible for handling control to other objects because they are deciding what needs to be done. In other terms we will call them as clients. Like may be some of you may have had the situation to deal with the client situation where the client is subscribing to a project pays you for the project and you have to provide service to the client.

Naturally you know that you don't decide when things will happen how things will happen? The client will decide that and based on that you will have to react. So concept of active object is very similar to that. In contrast passive objects are passively waits for messages to be processed. They by themselves do not start doing anything they by themselves do not change their state. They are just in threads which are just listening if there is some message from a client that is arriving so they get activated when they receive a message from another object.

So in this process once they receive the message they know what service needs to be provided to the client they just provide that service and accordingly the states of the passive objects get changed by what the active object send messages to them. So these are known as the server objects.

(Refer Slide Time: 22:46)

Synchronization

For objects with multiple threads of control, synchronization needs to ensure mutual exclusion to resolve issues of concurrency

Methods of concurrency used for a link between an active and a passive object:

- Sequential**
 - Passive object linked to a single active object at a time
 - Active objects must coordinate outside the object so that only one flow is in the object at a time (*function invocation*)
 - For multiple flows of control, the integrity is not guaranteed
 - One **Printer** (passive) – one **Print Task** (active)
- Guarded**
 - The invoking object (Client) coordinates for mutual exclusion
 - Integrity is guaranteed for multiple flows of control – serialize all calls to all of the object's guarded operations
 - Exactly one operation at a time can execute on the object, reducing this to sequential semantics (**Beware of deadlock!**)
 - One **Printer** – one **Print Task** uses a queue to serialize
- Concurrent**
 - The invoked object (Server) coordinates for mutual exclusion
 - Integrity is guaranteed for multiple flows of control – multiple flows access disjoint sets of data or only read data (**Crafty Design**)
 - One **Printer** uses a **Spooler** – multiple **Print Tasks**

So given that there is different situation that need to be synchronized I just mentioned that we need to understand what is synchronization all about? And what could be issues? Let us take a situation which I am sure many of you if not all of you would be familiar with suppose you are working in an organization or in your college there is a printer which is connected to the network. So I have what I have is a printer which is connected to the network. This is a network.

And different computers are connected C1 is one computer, C2 is another computer; C3 is another computer which are all connected to the network. These days we always have computers connected on the LAN or Wi Fi and so on. Now in these computers different independent people are working so they decide to send files for printing to the same printer. There is only one printer. So let us assume that C has requested to print file 1 and C2 has requested to print file 2 at the same time.

Let us say file 1 has 10 pages this has say 20 pages so both of these request arrive on the object print. Now you will understand that certainly every page on the file that needs to be printed must be printed in a consistent matter. Certainly you do not want that you print two pages of file 1 and then three pages of file 2 or if you are doing back to back printing the situation can get even worse. You on page you print a page of file 1 and in the back you print the page of file 2.

You would like to avoid that. But the request has come together and the request will need to be serviced by the printer at the same time. So we need to make sure that even in this situation where C1 the computer 1 requesting for the printer to print file 1 conceives as if the printer is exclusively available to C1. C2 at the same time requesting the printer object p to print file F2 also thinks that the print belongs only to C2.

C3 for now is not printing but C3 also thinks the same way. So we need to make sure that in this context things can happen correctly. That is all this request should not get mixed up. So the process by which we make sure that these printing tasks or printing file 1 and file 2 would be done in some way. The request can come in any arbitrary order but they must be serviced in a certain order so that result is consistent which means that either I print file 1 complete it.

Then I print file 2 or I print file 2 complete it then I print file 1 or if the request to print file 2 comes when I am already printing file 1 that request will somehow have to be put aside till I am finished with printing file 1. So what we need to do this is known as in terms of concurrency studies or in terms synchronization this is known as mutual exclusion. Mutual exclusion means that if there is more than one request which need to be service by same object at the same time then they will be serviced in certain serial order.

That is the two request will have to be mutually exclusive certainly also please note that not all request need to mutual exclusively. For example, if an object has a value and two other object send request to read that value that can happen simultaneously because it does not matter how many times you read a value from the same variable location you get the same value. But the situation will require mutual exclusion when you need to write that value at that place.

So that needs synchronization that needs mutual exclusion to be put in place and there are three basic strategies to do that for concurrency. One is sequential where there is a passive object that is providing the service and it is linked to only one active object. So the object decides when it is to send the request to the passive object and therefore passive object has nobody else the passive object will service.

So in the context of the example we were talking of that means that you have a PC and you have a printer. Your printer is directly connected to your PC there is no one else. So the print task the active object can use the printer object the passive one directly which this situation is like a function invocation as if the calling function and this is the called function. This is the simplest situation of synchronization which is sequential.

The second could be what is known as the guarded concurrency. What it does is the requesting, the active object the print task will know that well there are multiple files to print. So what it means that the computers C1 and C2 will not send the request directly to the printer but it will send it another object say a print task or print manager kind of object which having received two request can make some kind of queue between them based on the arrival and send them one by one to the printer.

So looking from the passive object printer it gets tasks one after the other whereas the print task which is an active object which actually received the request is managing how to put multiple request in a serial order so this print task is basically providing the guard that two request will not lay in the printer at the same time. So this is the situation a guarded concurrent is a situation where the invoking object or the client object manages the mutual exclusion.

And in that last case it could be that if you are aware most of the printers these days have an internal mechanism to queue up requests. These are called printer spoolers. So it could be that your printer is advanced enough to have a spooler so it does not matter the passive object printer could be given two request at the same time but it will internally do a spooling. It will internally doing a queuing and manage the sequencing of the two request.

So this is a situation where the invoked object in this case the printer or the server object actually manages the mutual exclusion and the multiple print tasks would just blindly shoot print request to the printer. If this situation exists, then we say it is a situation of concurrent access. So we have seen three kinds of concurrency sequential, guarded and concurrent.