



# Building the Class Diagram

## Building the Class Diagram

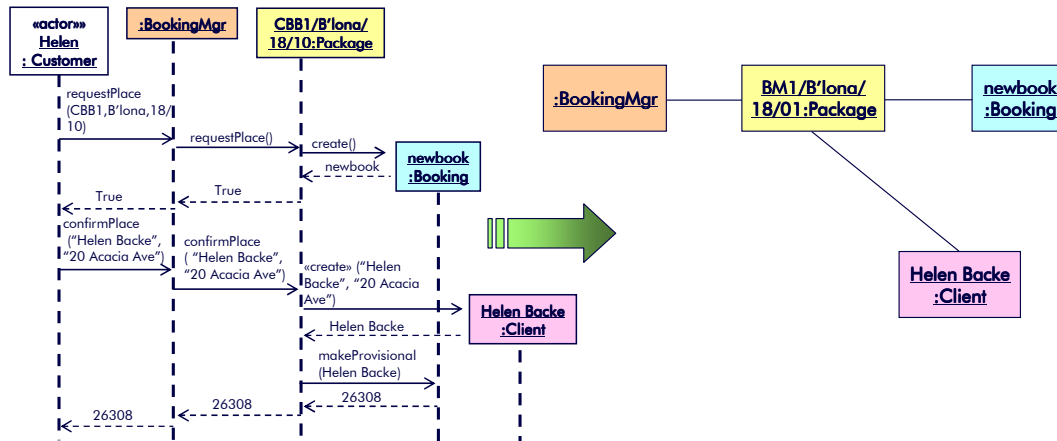
### Objectives

- To introduce the Communication Diagram
- To show how the Class Diagram can be built from the Sequence and/or Communication Diagrams.

# Communication Diagram

## Sequence Diagrams and Communication Diagrams are two views of the same scenario

- Sequence Diagram gives temporal view of a scenario
- Communication Diagram gives structural view



©tecademy

3

Communication Diagrams record the same information as sequence diagrams and, hence, scenarios. They just provide a different view – one that focuses on the structural view of the object interactions, rather than the temporal view. Some tools even generate communication diagrams from sequence diagrams (or *vice versa*) at the touch of a key.

Communication Diagrams appear, at first glance, to offer little to the Analyst that is not available from the sequence diagram. They are more often used by designers when planning for object interaction at run-time.

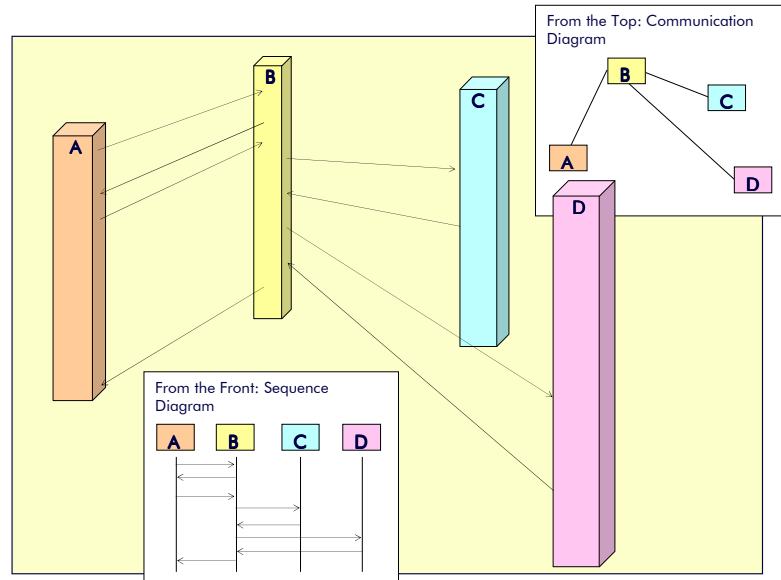
Many experienced Analysts, however, find that the more modelling they do, the more often certain arrangements of objects or classes seem to occur, and these are often associated with particular tasks. These arrangements become recognisable as *patterns*.

Patterns have become a very important part of the modeller's toolset, and an awareness of which kinds of patterns can be used to tackle which kinds of problems can often make the modeller's job a little more manageable.

It is easier to recognise these patterns in Communication Diagrams than in Sequence Diagrams. Communication Diagrams also help to define the shape of the Class Diagram, as we shall see..

# The Fence-post Model

## Two views of the same model



The transformation of a Sequence Diagram to a Communication Diagram becomes straightforward when it is recognised that both diagrams are projections of an underlying 3-dimensional model, as shown above.

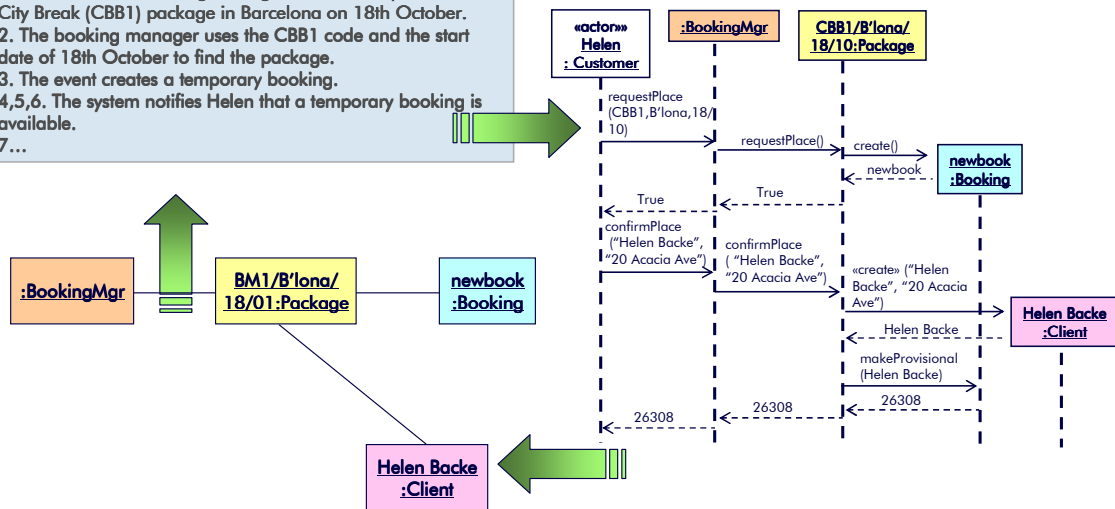
Imagine a fence, made of posts and wire. From the front, the view is not dissimilar to a Sequence Diagram. From above, however, the wires merge into each other and the structural arrangement of the fence-posts becomes more visible.

# Three of a Kind

## Scenario...

1. Helen asks the booking manager to reserve a place on the City Break (CBB1) package in Barcelona on 18th October.
2. The booking manager uses the CBB1 code and the start date of 18th October to find the package.
3. The event creates a temporary booking.
- 4,5,6. The system notifies Helen that a temporary booking is available.
- 7...

## ... to Sequence Diagram...

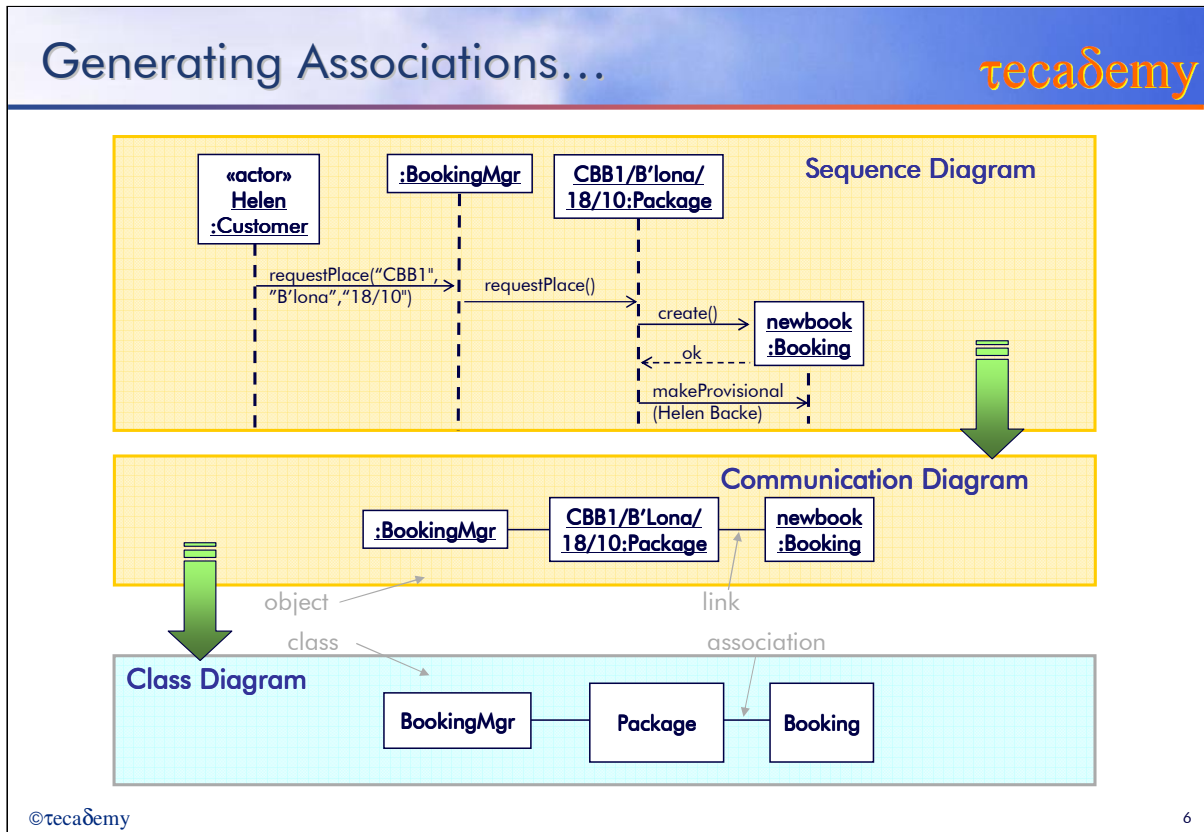


## ... to Communication Diagram...

Communication Diagrams, therefore, give the structural view of a scenario. Remember, there is a 1:1:1 relationship between a Scenario, Sequence Diagram and a Communication Diagram, so it is valid to say that if a scenario documents a single, detailed requirement, then so does a communication diagram.

We can use this 1:1:1 relationship to add details our class diagram that will link it to every requirement we have been given to model.

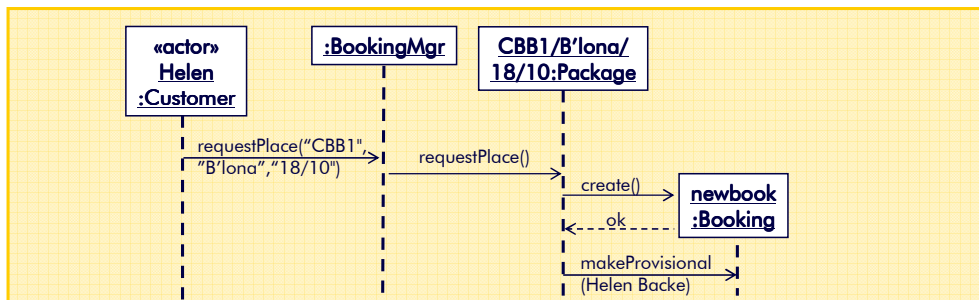
We now turn our attention to this process.



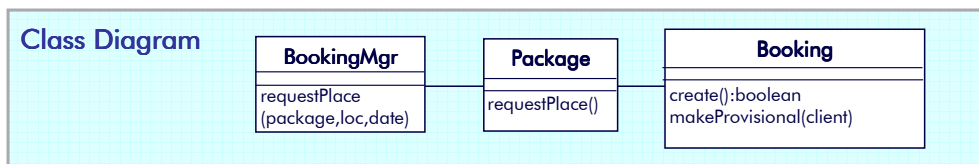
An association signifies a relationship between classes. Whenever an object in a sequence diagram sends a message to another object, there must be an association between their classes in the class diagram. (with the exception of a create message, this only signifies a temporary link)

The tentative class diagram created earlier, was created without regard to the sequence diagram, this model will now need changing, missing classes and associations will need to be added and existing ones may need changing. The model should be checked to determine whether everything was really necessary and whether some functionality has been missed.

## Generating Operations



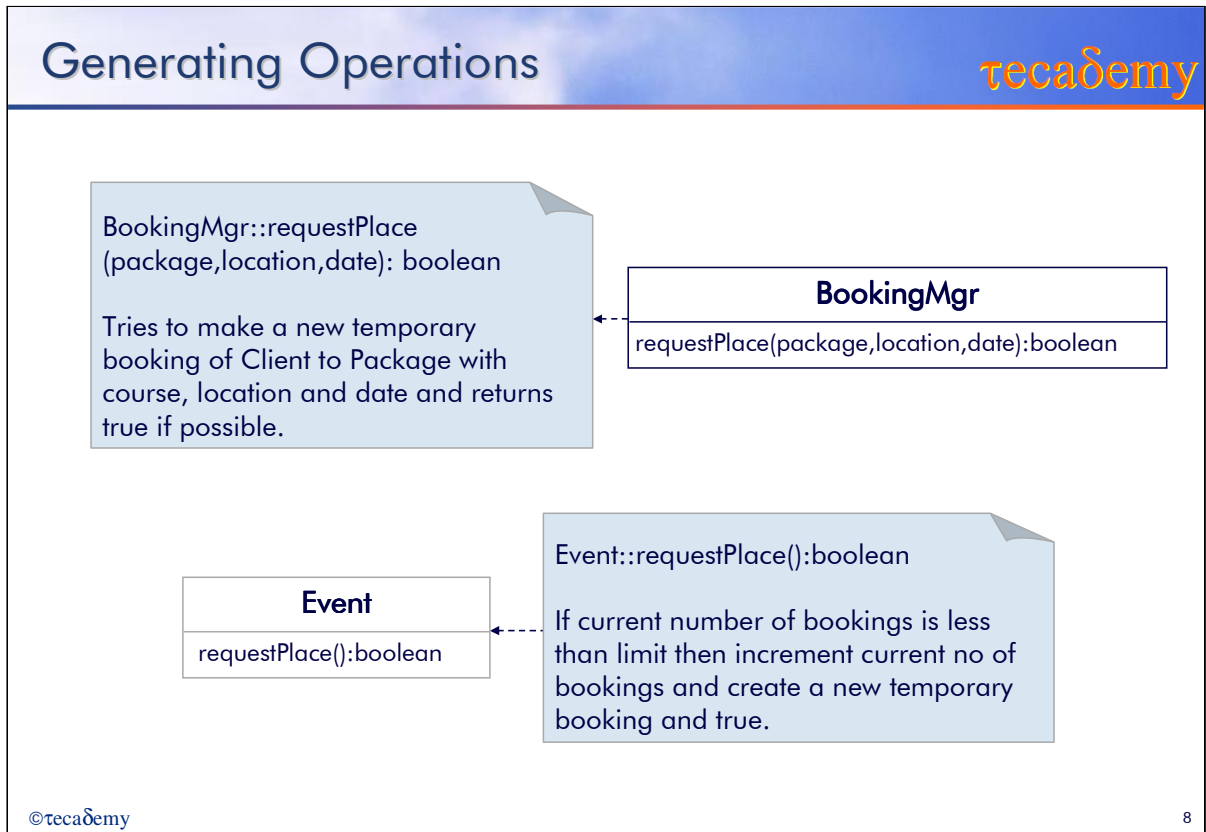
- A message identifies an operation on the receiver
- A return is handled as a return type



A message in a sequence diagram is synonymous with the sending object demanding that the receiving object performs the named task. The receiver must then have an operation to implement the order. A return is mapped as the return of the operation, and thus does not cause a new operation. Instead, a return is noted as a return type on the initial operation.

The operation should be described as best as we can at this stage, as the sequence diagram contains the context in which the operation is being invoked. When the operation is in the class diagram, the context will be lost (or at least, not easily available without maintaining backwards traceability).

Some systems allow non-procedural code, such that the sender does not have to wait for the receiver to return. A 'return' message in such a system is therefore not necessarily a procedural return, but instead, just a logical message from the original receiver to the original sender. In this case, messages in both directions are treated equally, as operations. Such a system would also allow a different object (or none) to produce a logical return. Sequence diagrams for this approach are allowed in the UML, and differ from the ones discussed in the lectures. They usually omit control regions and procedural returns, and all messages would be mapped to operations on the class diagram.

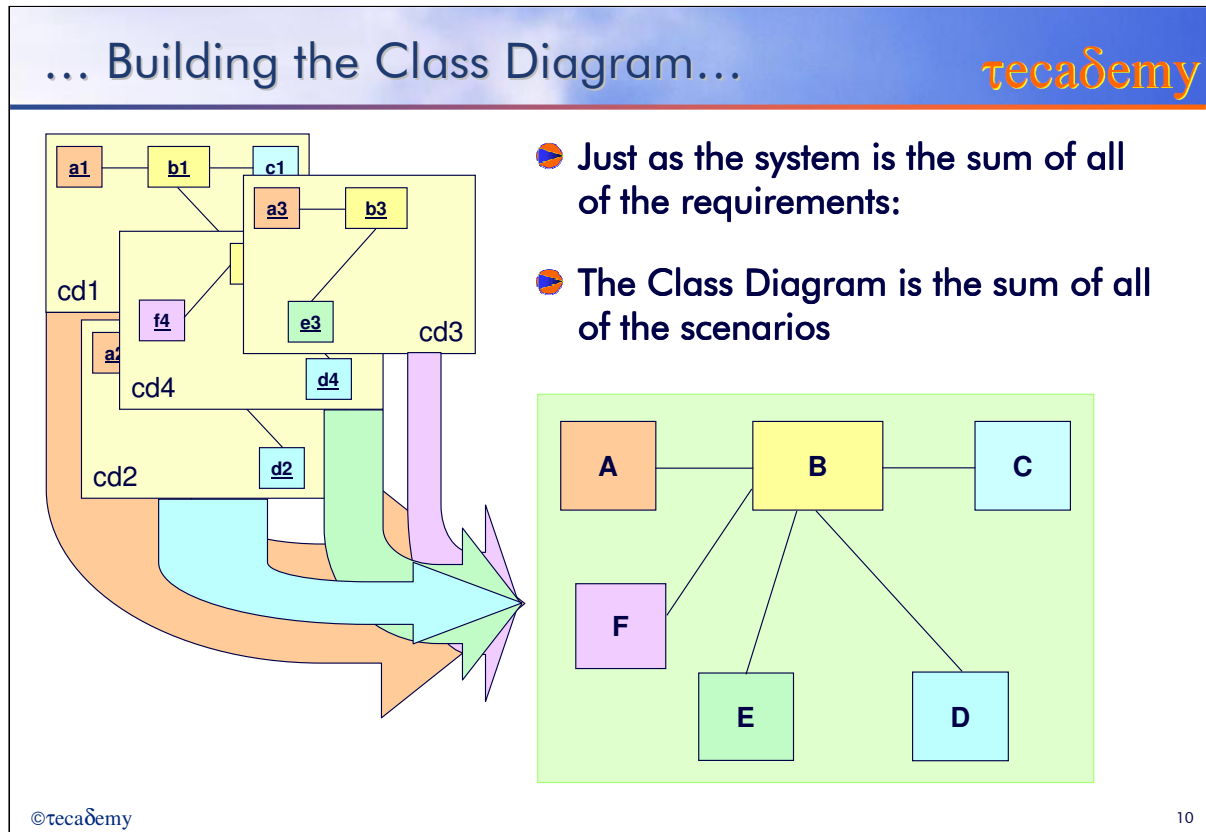


Each operation may be specified in the repository. This ensures agreement about what it is that an operation does.

## Operation Support

- ▶ **Need to decide what information an object requires to perform an operation**
- ▶ **Refer to the class diagram**
  - ▶ It may already have enough information in its attributes
  - ▶ It may be able to find information through its associations
- ▶ **May need to add support**
  - ▶ Add attributes and/or associations if the receiving object needs permanent knowledge
  - ▶ Add parameters if the knowledge is temporary
    - ▶ Temporary means during one operation call
    - ▶ Permanent means over more than one call

When deciding how far to take the sequence diagram in terms of number of objects and complexity of operations, we should have considered how an object will be able to perform the operation. This information now needs to be recorded. We need specifically to identify what knowledge the object requires. Certainly, in early sequence diagrams, this may be left unspecified; a message can be sent to an object that is thought to be responsible for performing the operation, and it can be left at that for a while. However, the analysis is not complete until we have captured the detail of how an object could perform its operation.



Which brings us to the following line of reasoning:

**If:**

A scenario is a detailed requirement for one case of system behaviour.

The system is the sum of all of the scenarios supported.

Each object on a communication diagram will be represented by a class on a class diagram.

Each link on a communication diagram will be represented by an association or an aggregation on a class diagram.

**Then:**

The class diagram (of the system) is the sum of all of the communication diagrams (of the scenarios).

This means that a first-cut class diagram can be constructed by:

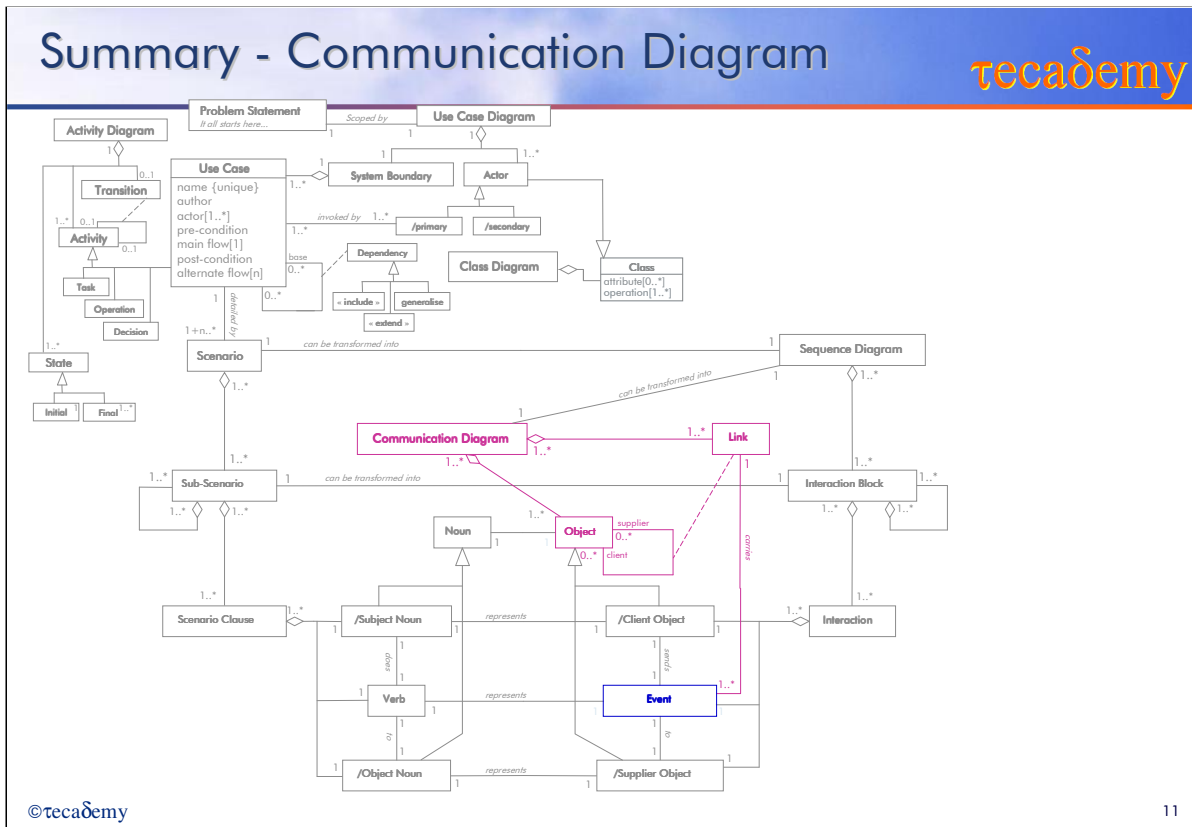
Drawing every scenario's communication diagram on a sheet of clear foil (such as those used on overhead projectors), so that objects of the same class occupy the same position on each foil;

Overlaying the foils and viewing the resultant diagram through the foils;

Replace each stack of objects with a class, and each link with an association;

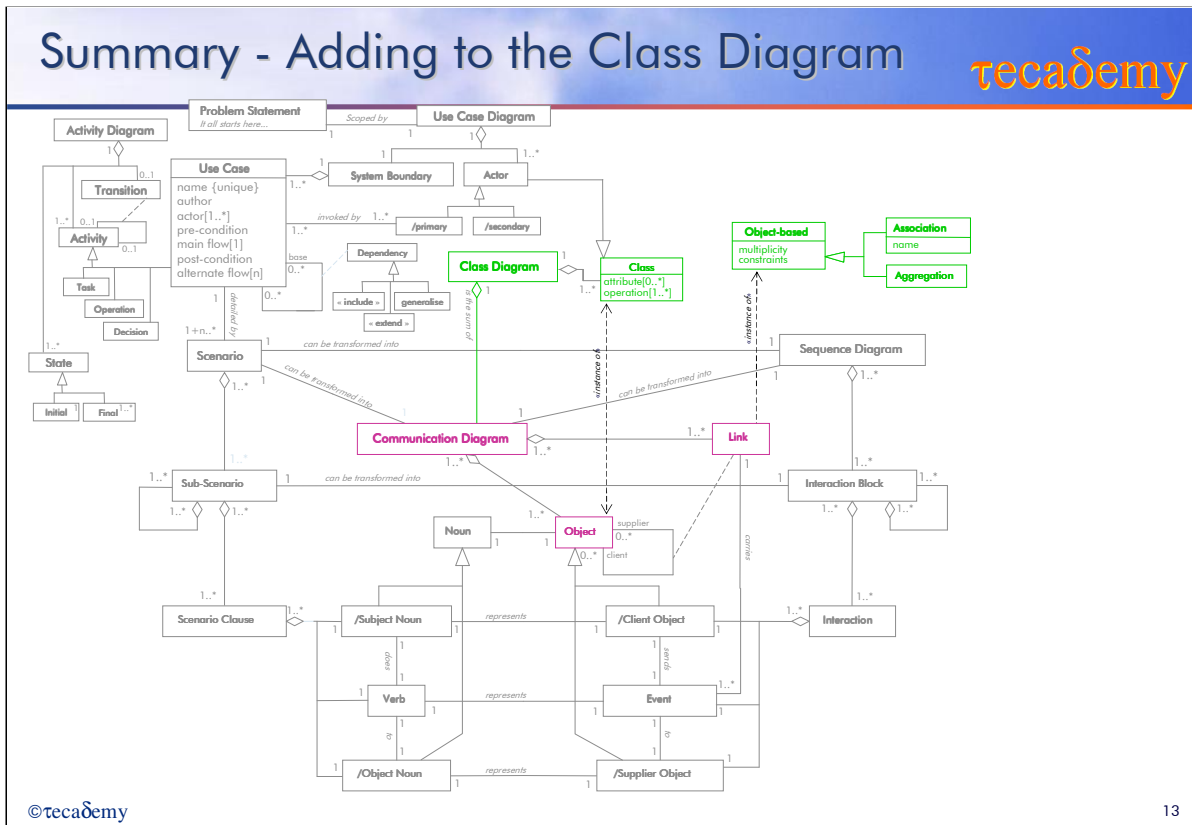
Move every message onto the target object's class as an operation signature.

Well, it's a start – there are still relationships to be named, multiplicities assigned, roles labelled, aggregations decided on... but the important thing is that we are working from a model that is derived *from the scenarios*. This in turn means we can justify every aspect of the model in terms of the stated requirements – a good position to be in.



The Communication Diagram shows objects and the links between them. The links carry messages between the objects.





Each object on a communication diagram must map to a class on the class diagram; each link to an object-based relationship, either an association or an aggregation. It follows that the class diagram can be thought of as being made up of all the possible communication diagrams – i.e. every element of the class diagram can be traced back to an individual scenario or requirement.

## Case Study - Exercise 10

### ▶ Objectives

- ▶ To practice updating a class diagram using sequence diagrams
- ▶ To consolidate understanding of the relationship between sequence diagrams and class diagrams

▶ Turn to and complete exercise 10 in the exercise booklet