

A large red circle is positioned in the top-left corner. Below it, a light blue shape with a curved top edge extends downwards and to the right, partially overlapping the text area.

Communication Diagrams

Massimo Felici

JCMB-1402 0131 650 5899

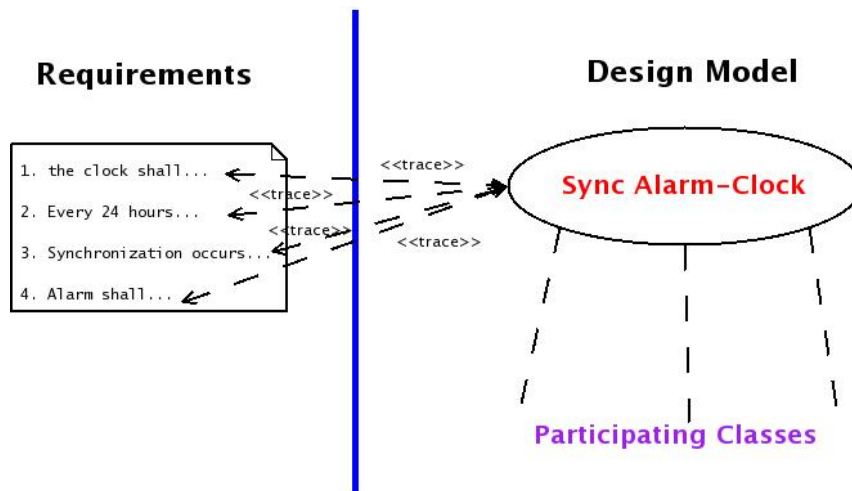
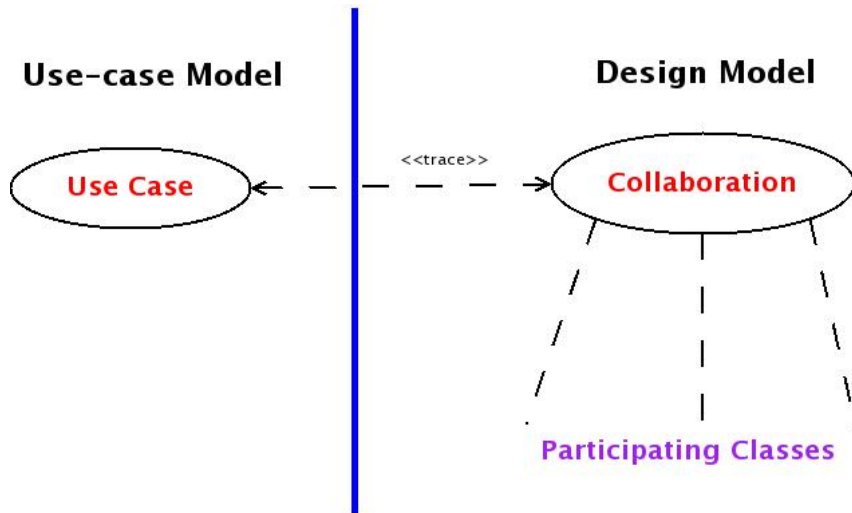
1BP-G04 0131 650 4408

mfelici@inf.ed.ac.uk

Communication Diagrams

- The **communication** is implicit in a **Sequence Diagram**, rather than explicitly represented as in a **Communication Diagram**
- There is some redundancy between **Communication** and **Sequence Diagrams**
 - They differently show how elements interact over time
 - They document in detail how classes realize user cases
 - **Communication Diagrams** show relationship between objects
 - **Sequence Diagrams** focus on the time in which events occur

Realizing Use cases in the Design Model



- **Use-case driven design** is a key theme in a variety of software processes based on the UML
- UML supports specific modeling constructs that realize use cases in the implementation
- **Collaborations (Communications)** enhance the systematic and aggregate behavioral aspects of the system
- Collaborations support **traceability** from requirements expressed in use cases into the design

What is a Collaboration?

- A **Collaboration** is a collection of named **objects** and **actors** with **links** connecting them
- A **Collaboration** defines a set of participants and relationships that are meaningful for a given set of purposes
- A **Collaboration** between objects working together provides emergent desirable **functionalities** in Object-Oriented systems
- Objects collaborate by **communicating** (passing messages) with one another in order to work together

Collaborations

■ Actors

- Each Actor is named and has a role
- One actor will be the **initiator** of the use case

■ Objects

- Each object in the collaboration is named and has its class specified
- Not all classes need to appear
- There may be more than one object of a class

■ Links

- Links connect objects and actors and are instances of associations
- Each link corresponds to an association in the class diagram

Interactions

- Use cases and Class Diagrams constrain interactions
- **Associations** and **Links** in a **Collaboration Diagram** show the paths along which **messages** can be sent from one instance to another
- A **message** is the specification of a **stimulus**
- A **stimulus** represents a specific instance of sending the **message**, with particular arguments

Communication Diagrams' Rationale

- Model collaborations between **objects** or **roles** that deliver the **functionalities** of use cases and operations
- Model mechanisms within the **architectural design** of the system
- Capture interactions that show the passed **messages** between objects and roles within the collaboration
- Model **alternative scenarios** within use cases or operations that involve the collaboration of different objects and interactions
- Support the identification of objects (hence classes) that participate in use cases

Communication Diagrams

- **Specification level** shows generic cases of collaborations (communications)
 - **Generic form** captures a collaboration among class roles and association roles and their interactions
- **Instance level** shows a specific instance of an interaction taking place and involving specific object instances
 - **Instance form** captures a scenario among objects conforming to class roles and links conforming to association roles

Lifelines and Links

- **Participants** on a collaboration diagram are represented by a rectangle
- The syntax for the name of a lifeline



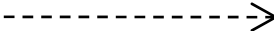
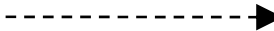
[connectable-element-name][['selector']][:class-name][decomposition]

- A **communication link** is shown with a single line that connects two participants



Messages

- A **message** on a communication diagram is shown using an arrow from the message sender to the message receiver
- **Message Signature**
return-value, message-name, argument-list

			
Synchronous Or Call	Asynchronous	Creation	Reply (Return)



Messages

- **Procedural** or **Synchronous**: A message is sent by one object to another and the first object waits until the resulting action has completed.
- **Asynchronous**: A message is sent by one object to another, but the first object does not wait until the resulting action has completed.
- **Flat**: Each arrow shows a progression from one step to the next in a sequence. Normally the message is asynchronous.
- **Return**: the explicit return of control from the object to which the message was sent.
- Messages occurring at the same time: Adding a **number-and-letter** notation to indicate that a message happens at the same time as another message
- Invoking a message multiple times: **Looping constraint**, e.g., `*[i=0..9]`
- Sending a message based on a condition: A **guardian condition** is made up of a logical boolean statement, e.g., `[condition=true]`
- When a participant sends a message to itself



Where should messages go?

- The message is directed from sender to receiver
- The receiver must understand the message
- The association must be navigable in that direction
- Law of Demeter
- Dealing with a message m an Object O can send messages to:
 - Itself
 - Objects sent as argument in the message m
 - Objects O creates in responding to m
 - Objects that are directly accessible from O , using attribute values



Activations: Flow of Control

■ Procedural interactions

- At most one object is computing at any time

■ Activation

- An object has a live activation from when it receives a message until it responds to the message

■ Waiting for response

- Synchronous messages on sending a message to another object, an object will wait until it receives a response

■ Activation task

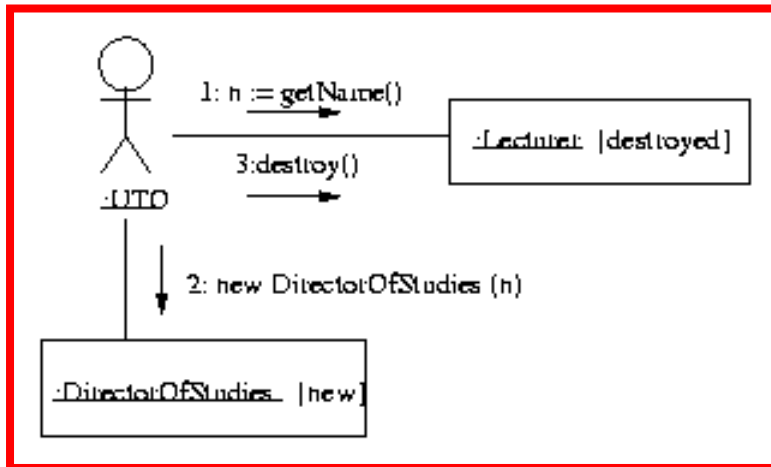
- Activations are stacked and the top activation has control. When the top action responds the next to top regains control and so on...



Creation and Deletion

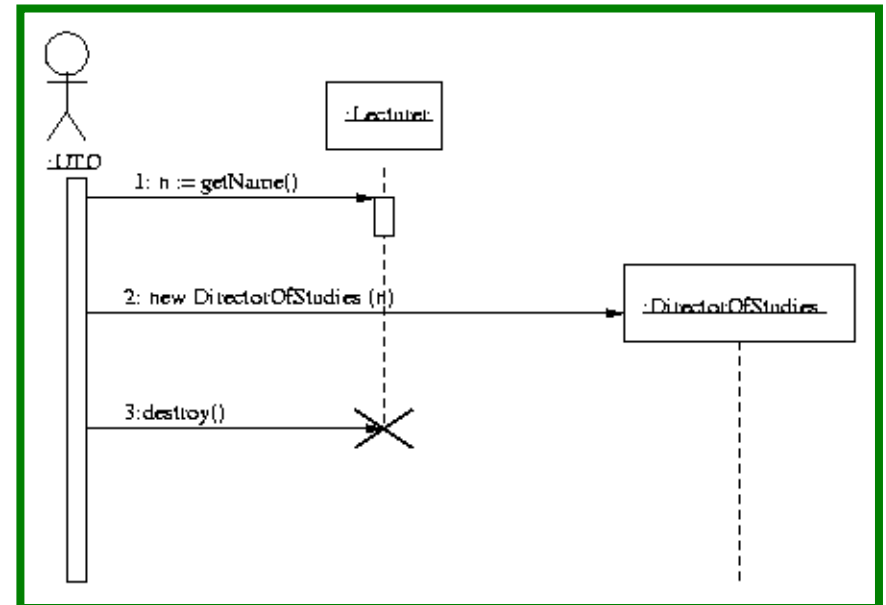
In **Communication Diagrams** the objects are labeled:

- New for objects created in the collaboration
- Destroyed for objects destroyed during the collaboration



In **Sequence Diagrams**, It is possible to use the lifelines

- New objects have their icon inserted when they are created
- Destroyed objects have their lifeline terminated with **X**



Communication vs. Sequence Diagrams

	Communication Diagrams	Sequence Diagrams
Participants	✓	✓
Links	✓	
Message Signature	✓	✓
Parallel Messages	✓	✓
Asynchronous messages		✓
Message Ordering		✓
Create & Maintain	✓	



Constructing Communication Diagrams

1. Identify **behavior**
2. Identify the **structural elements**
3. Model **structural relationships**
4. Consider the **alternative scenarios**



Readings

- **UML course textbook**
 - Chapter 10 on More on Interaction Diagrams



Summary

- Interaction Diagrams
 - Sequence Diagrams
 - Communication Diagrams
- Communication Diagrams' Rationale
- Communication Diagrams
 - Collaborations
 - Interactions
 - Messages
- Constructing Communication Diagrams

