Chapter 6. Process Modelling with UML

Reyes Grangel Seguer

EI1030. Analysis of Inf. Systems / EI1032. Software Analysis / MT1046. Integrated Inf. Systems
Computer Engineering / Computational Mathematics
Dept. of Computer Systems and Languages

March 20, 2019

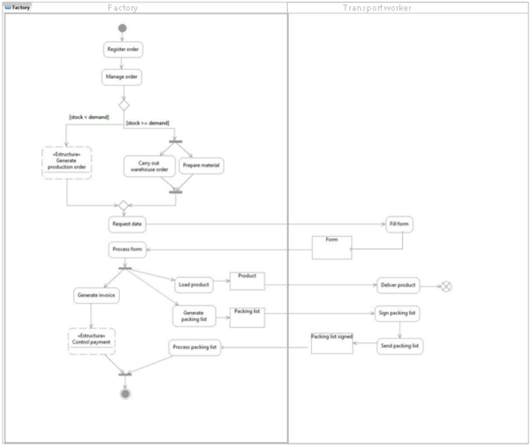
Outline

Outline

Objective

- To show connections between actions to understand complex behaviours
- It is useful for representing the behaviour of a ...
    - business process
    - use case
    - method
    - etc.

Example: Activity Diagram



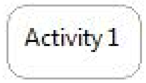
Constructs

- Activity
- Transition
- Control nodes

Activity

What does it represent?

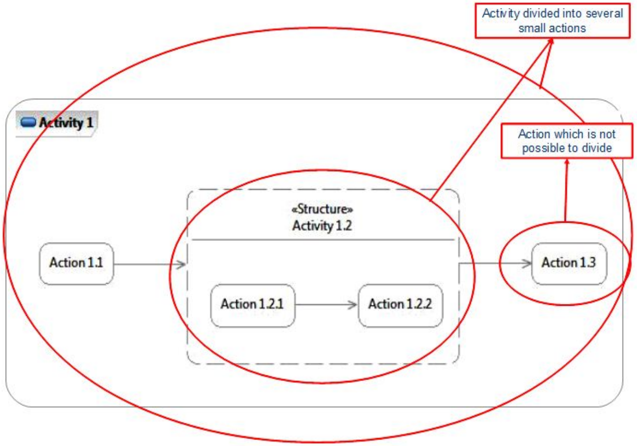
The execution of a set of sequential actions which define a workflow



Differences between ...

Activity	Action
Non atomic execution	Atomic execution
Can be divided into actions	Indivisible
Complex behaviour	Simple behaviour, for example
	One call to another operation
	The submission of one signal

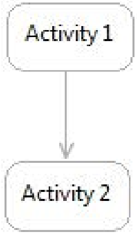
Example: Activity vs Action


Activity divided into several small actions
Action which is not possible to divide
Activity 1
«Structure»
Activity 1.2
Action 1.1
Action 1.2.1
Action 1.2.2
Action 1.3

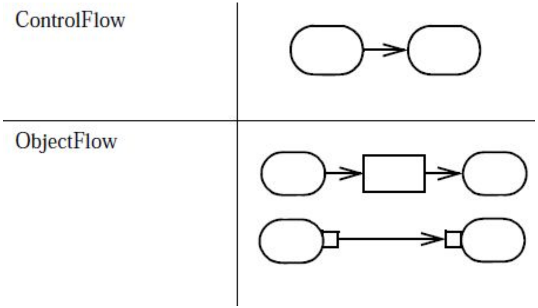
Transition

What does it represent?

The step of one action to another one, automatically the end of one action sparks the transition, which produce the beginning of another activity



Transition

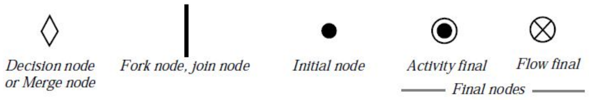


- Control Flow
- Object Flow

Control nodes

What does it represent?

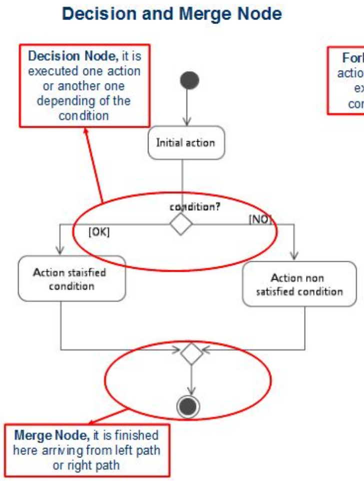
The coordination of the flows between other nodes

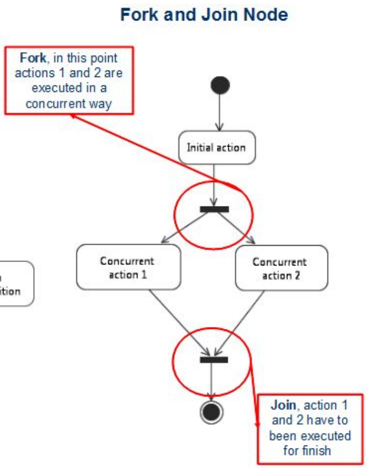


Types

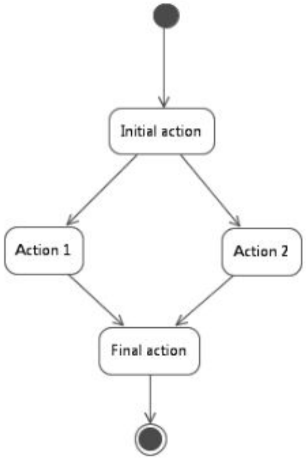
- Decision node and merge node
- Fork node and join node
- Initial node
- Final node
    - Activity final
    - Flow final

Differences between control nodes ...


Decision and Merge Node
Decision Node, it is executed one action or another one depending of the condition
Initial action
condition?
[OK]
[NO]
Action staisfied condition
Action non satisfied condition
Merge Node, it is finished here arriving from left path or right path


Fork and Join Node
Fork, in this point actions 1 and 2 are executed in a concurrent way
Initial action
Concurrent action 1
Concurrent action 2
Join, action 1 and 2 have to been executed for finish

... and in the following case



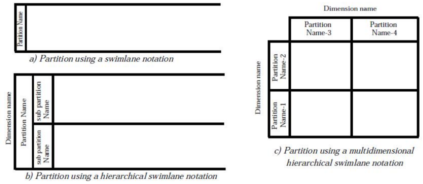
Activity Partition

What does it represent?

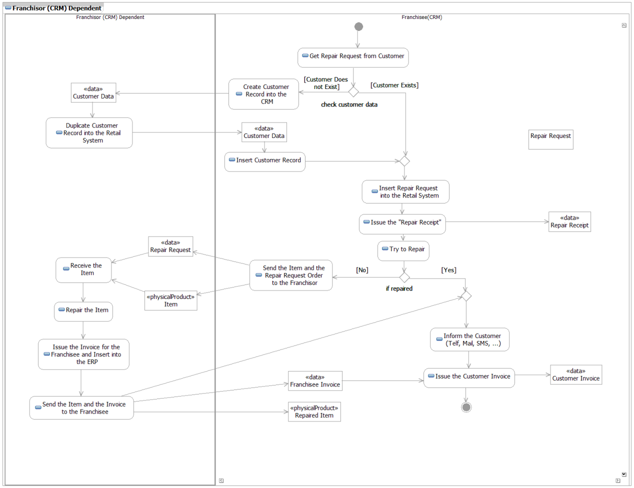
A kind of activity group for identifying actions that have some characteristic in common

Notation

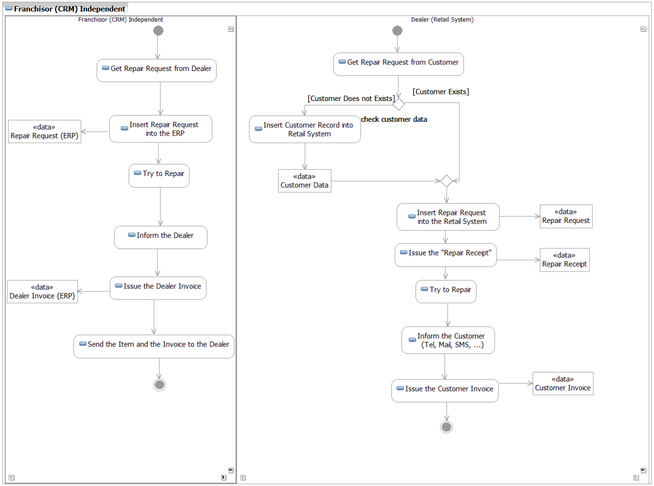
- Named also 'partitions'
- Named 'swimlanes' in previous versions
- They can be horizontals, verticals, or both
- They often correspond to organisational units in a business model



Example: Activity Diagram (Partitions)


Franchisor (CRM) Dependent
Franchisor (CRM) Dependent
Franchisee(CRM)
Get Repair Request from Customer
[Customer Does not Exist]
[Customer Exists]
Create Customer Record into the CRM
check customer data
«data» Customer Data
Duplicate Customer Record into the Retail System
«data» Customer Data
Insert Customer Record
Repair Request
Insert Repair Request into the Retail System
Issue the "Repair Receipt"
«data» Repair Receipt
Try to Repair
«data» Repair Request
Receive the Item
Send the Item and the Repair Request Order to the Franchisor
[No]
[Yes]
if repaired
«physicalProduct» Item
Repair the Item
Inform the Customer (Telf, Mail, SMS, ...)
Issue the Invoice for the Franchisee and Insert into the ERP
«data» Franchisee Invoice
Issue the Customer Invoice
«data» Customer Invoice
Send the Item and the Invoice to the Franchisee
«physicalProduct» Repaired Item

Example: Activity Diagram (Partitions)


Franchisor (CRM) Independent
Franchisor (CRM) Independent
Get Repair Request from Dealer
«data» Repair Request (ERP)
Insert Repair Request into the ERP
Try to Repair
Inform the Dealer
«data» Dealer Invoice (ERP)
Issue the Dealer Invoice
Send the Item and the Invoice to the Dealer
Dealer (Retail System)
Get Repair Request from Customer
[Customer Does not Exists]
[Customer Exists]
check customer data
Insert Customer Record into Retail System
«data» Customer Data
Insert Repair Request into the Retail System
«data» Repair Request
Issue the "Repair Receipt"
«data» Repair Receipt
Try to Repair
Inform the Customer (Tel, Mail, SMS, ...)
Issue the Customer Invoice
«data» Customer Invoice

Outline

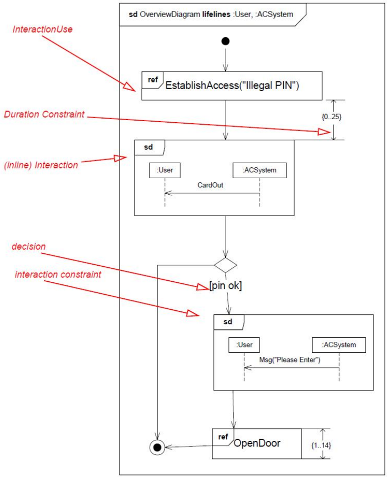
Types

- Sequence Diagram
- Communication Diagram
- Interaction Overview Diagram
- Timing Diagram

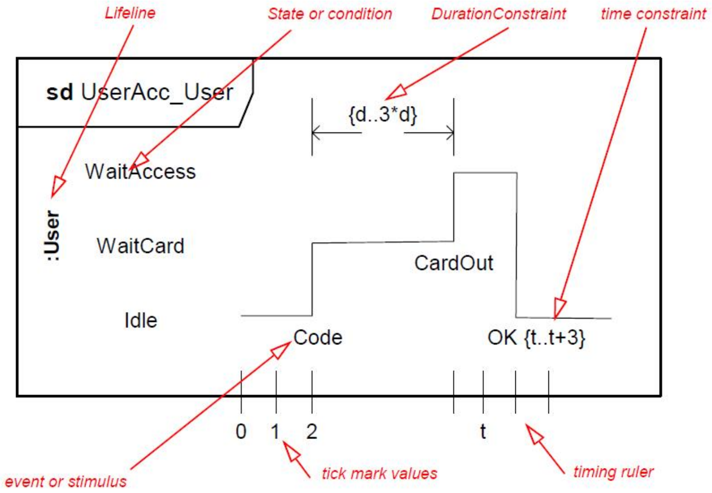
Features

- **Objective**: to show ...
  - the **flow of messages** for a specific scenario
  - **how objects interact** for executing a task
- So, they are performed at **instance level**
- Useful to **refine** operations and associations between classes
- Only to do for the most **complex scenarios** of difficult use cases
- Sequence and communication diagram are **isomorphous**
- So, it is possible to obtain **automatically** one from the previous one

Example: Interaction Overview Diagram



Example: Timing Diagram



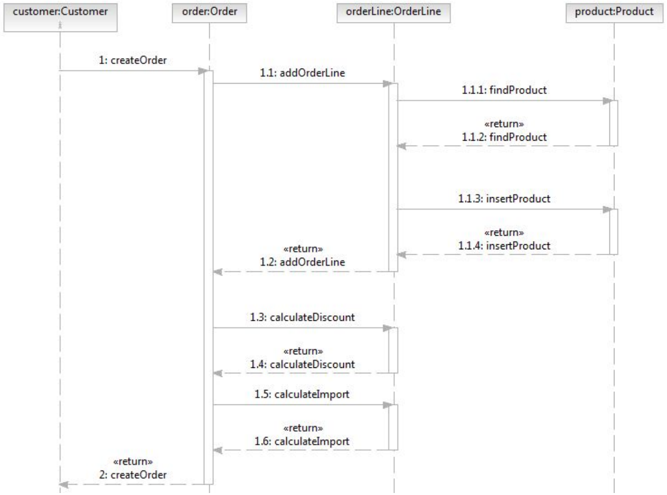
Features

- The actor who **trigger** the actions is the first element at the bottom left corner
- Appears **actors** (instance) and **objects** that interact in an scenario
- The **lifeline** of objects represents time from object viewpoint
- The **time** runs from top to bottom and from left to right
- A **message** is represented by an arrow from sender's lifeline to receiver's lifeline
- **Objects' order** is not compulsory, but is welcome

Basic Elements

- **Actor (instance)**: primary actor's instance related to the use case, who trigger the action
- **Object**: instance of one class, which participate in the interaction
- **Message**: communication establishes between objects in order to interact

Example: Sequence Diagram



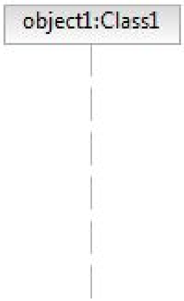
Lifeline

What does it represent?

The existence of an object down time period

Notation

- Vertical dashed line
- The most of the objects exists during the interaction
- To create objects during interaction: «create»
- To destroy objects during interaction: «destroy»



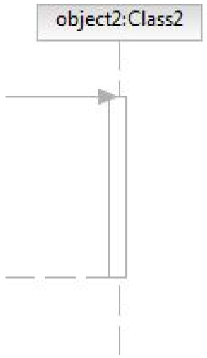
Execution Specification ('Foco de control')

What does it represent?

A time period in which an object executes an action, directly or subordinately

Notation

- Tin and vertical rectangle (grey or white) draw over lifeline
- The top of the rectangle is aligned with the beginning of the action
- And the bottom with the end
- It is possible to represent recursive calls

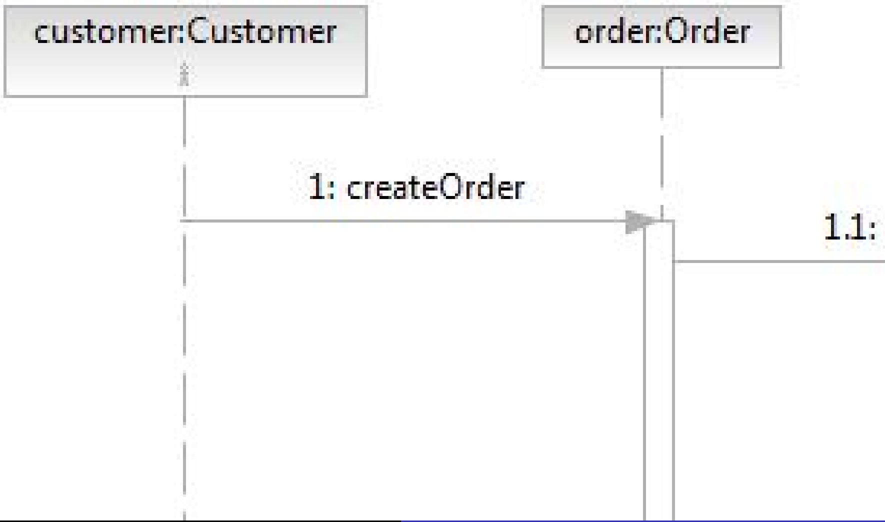


Conditions for Sending Messages

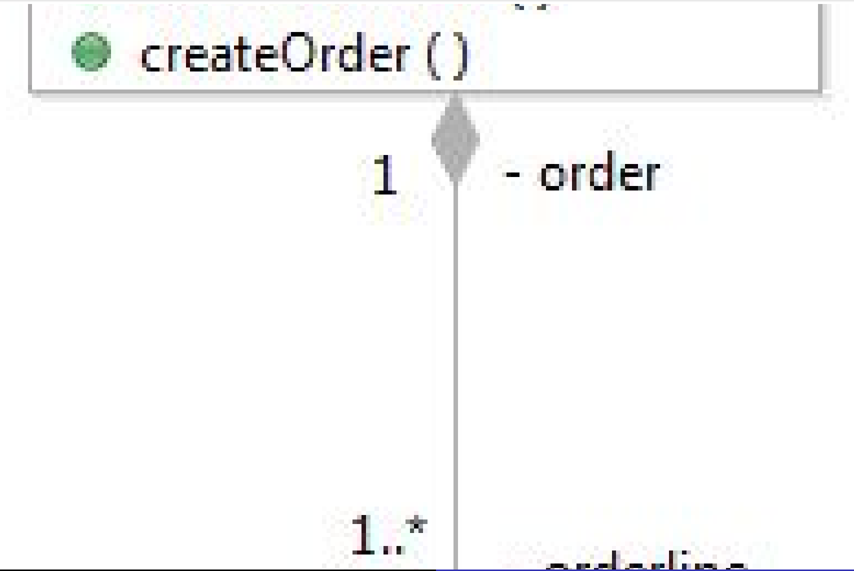
When one object 1 send one message X to another object 2 ...

1. It must exist a link between object 1 and object 2
2. It must exist an association between the corresponding classes
3. This association must be navigable in the sense of the submission of the message
4. The receiver class must have an operation able to understand message X
5. That is to say, the class 2 provide the appropriate operation X

Example: Sequence Diagram



Example: Class Diagram



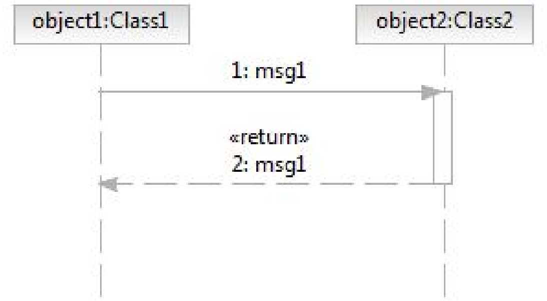
Types of messages

1. Synchronous message
2. Asynchronous message
3. Recursive message (synchronous or asynchronous)
4. «create»
5. «destroy»

Synchronous message

Notation

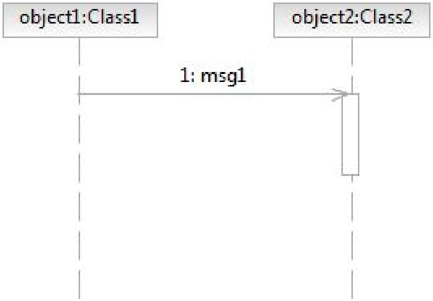
- The call is represented by filled arrow head
- The return is represented by a dashed line
- When the call is performed, the object, which does the call must wait for the return



Asynchronous message

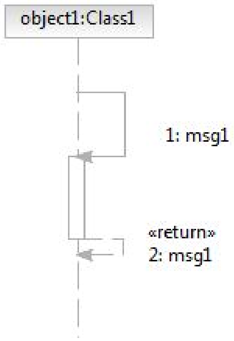
Notation

- It is represented by an open arrow head
- It is used in concurrent executions, when there is more than one thread of execution

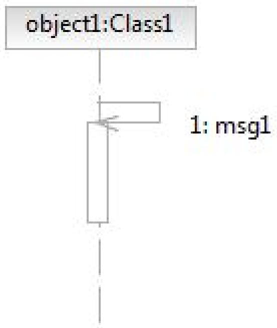


Recursive Messages

Synchronous recursive message

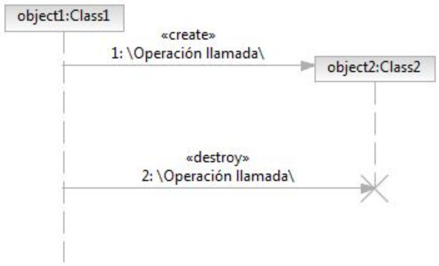


Asynchronous recursive message

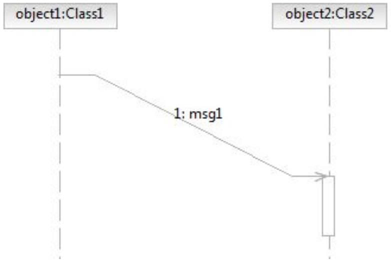


Other Kind of Messages

- **Create message**: dashed line with an open arrow
- **Destroy message**



Delay message



Combined Fragment (or Interaction Frame)

What does it represent?

An specific interaction fragment for describing a number of traces in a compact and concise manner

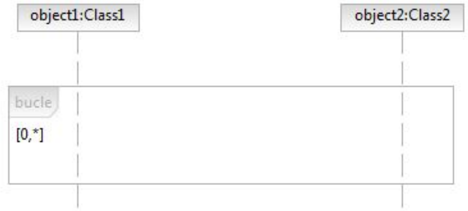
Notation

Its semantics is dependent upon the interaction operator explained in the next slide

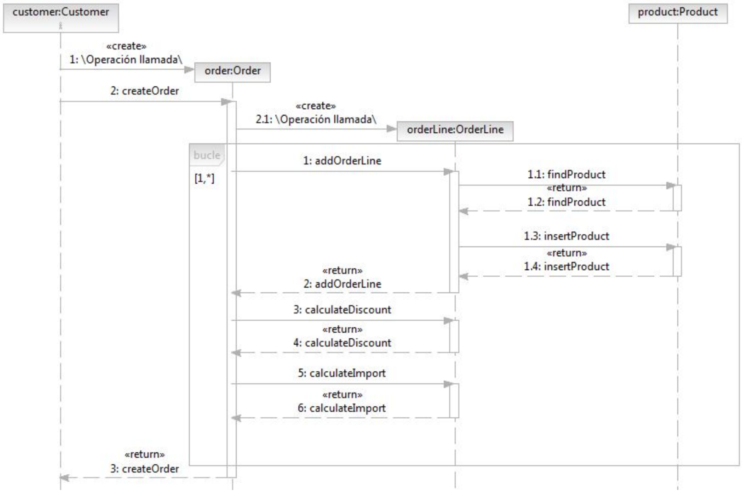
Combined Fragment (or Interaction Frame)

Notation

- It is defined by an interaction operator and corresponding interaction operands
- Possibles operations and its notation
  - alt: Alternatives
  - opt: Optional
  - par: Parallel
  - neg: Negative
  - loop: Loop
  - ...



Example: Sequence Diagram with CF



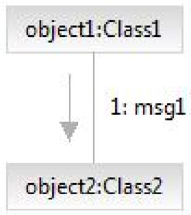
Basic Elements

- **Actor (instance)**: primary actor's instance related to use case, who trigger the action
- **Object**: instance of one class, which participate in the interaction
- **Message**: communication establishes between objects in order to interact
- **Link**: connection between objects which indicates that exists an association between corresponding classes

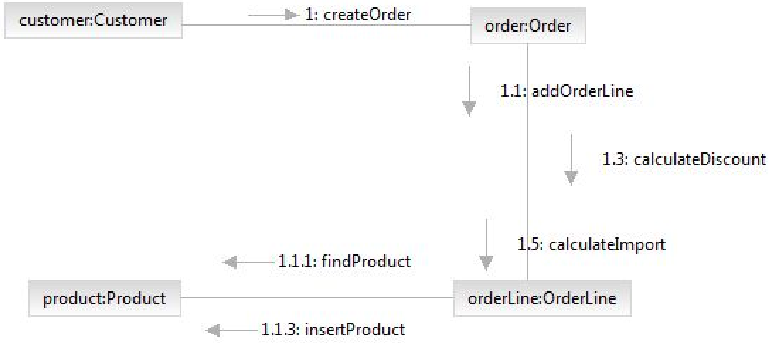
Message

Notation

- Represented by a small arrow in the direction of the message
- It is placed close to the message name and sequence number along the line between lifelines (objects)



Example: Communication Diagram


customer:Customer
1: createOrder
order:Order
1.1: addOrderLine
1.3: calculateDiscount
1.5: calculateImport
1.1.1: findProduct
product:Product
orderLine:OrderLine
1.1.3: insertProduct