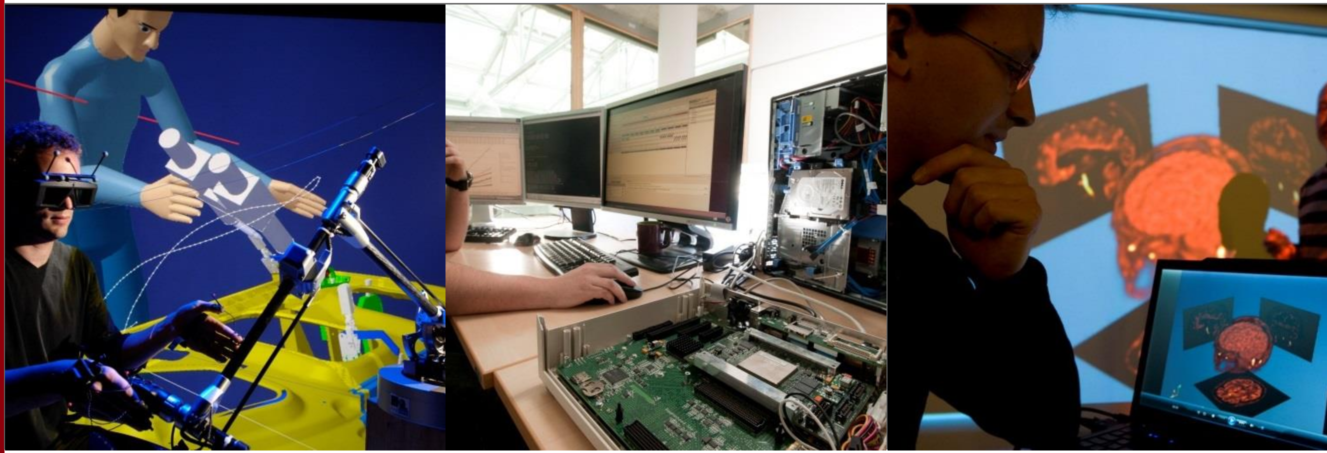
INTERNAL STRUCTURE MODELING

Introduction on UML for Industrial Systems

Shuai Li, Jérémie Tatibouët, François Terrier, Sébastien Gérard, **Asma Smaoui**

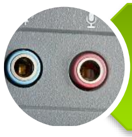
{first_name}.{last_name}@cea.fr



CONTENT

Introduction

Basic Elements


Port

Connector

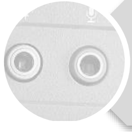
Quiz

Summary

CONTENT

Introduction

Basic Elements



Port

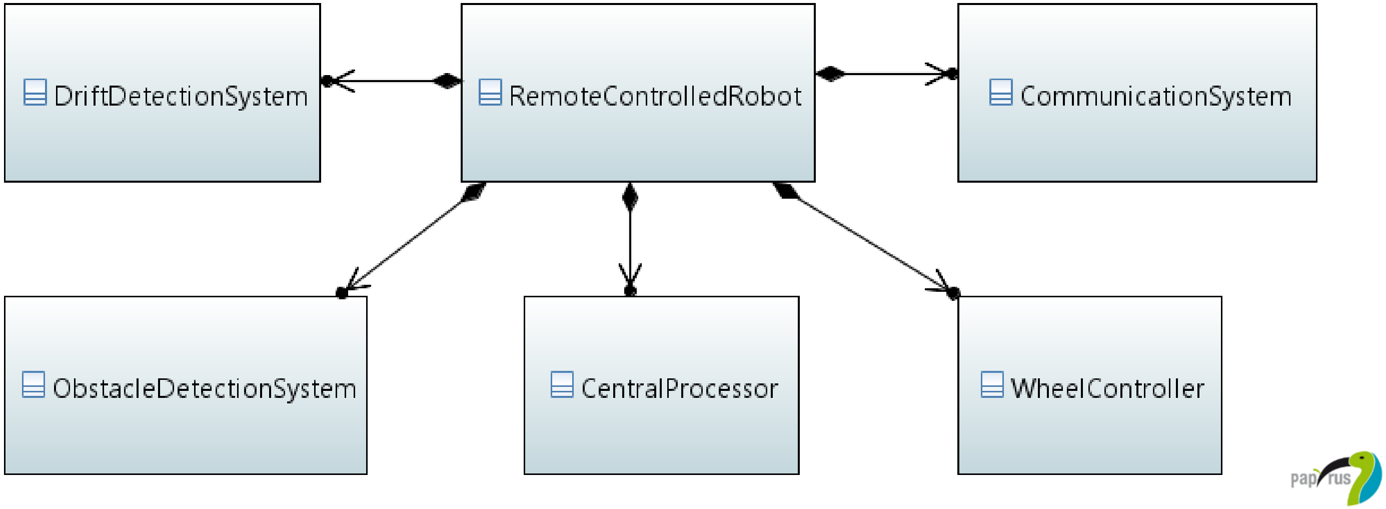
Connector

Quiz

Summary

Why model the internal structure of a class?

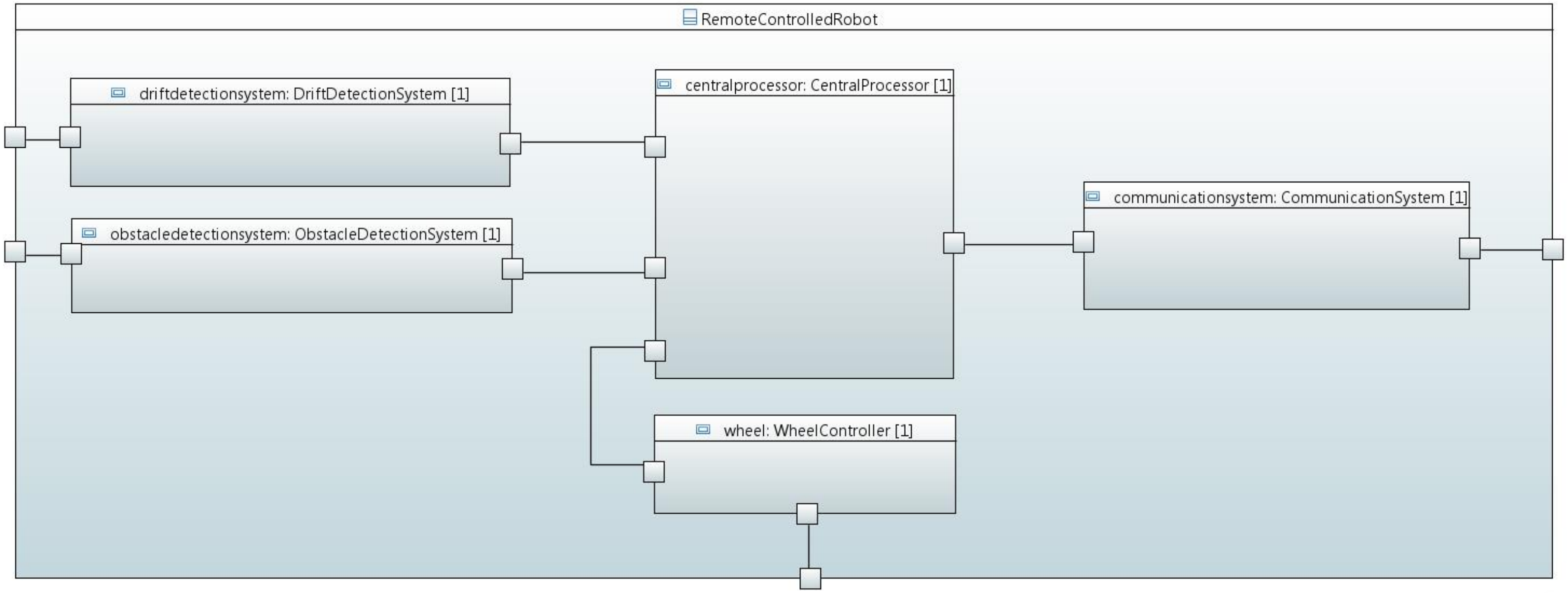
- We've seen how to model classes and their features; we've seen that objects can be related in a composition, i.e. whole/part relationship
- Modeling the internal structure is useful to describe the collaboration between interacting parts of a composite



❓ Can you see the collaboration between the parts of this system?

Why model the internal structure of a class?

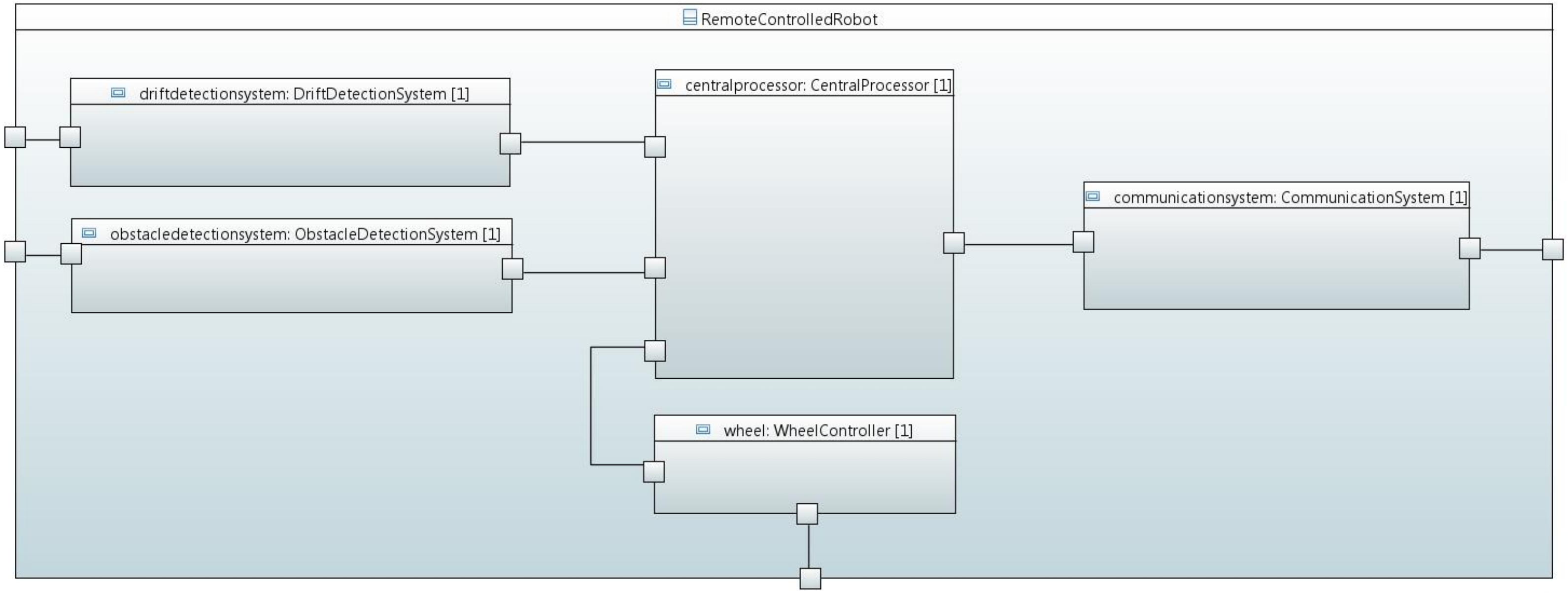
- **We’ve seen how to model classes and their features; we’ve seen that objects can be related in a composition, i.e. whole/part relationship**
- **Modeling the internal structure is useful to describe the collaboration between interacting parts of a composite**


RemoteControlledRobot
driftdetectionsystem: DriftDetectionSystem [1]
centralprocessor: CentralProcessor [1]
communicationsystem: CommunicationSystem [1]
obstacledetectionsystem: ObstacleDetectionSystem [1]
wheel: WheelController [1]

! What about now?

Internal structure modeling in UML

- The diagram below is a composite structure diagram
- The internal structure of a class can be described with composite structure diagrams ➔ we describe a composite structure
- Composite structure diagrams are new to UML 2


RemoteControlledRobot
driftdetectionsystem: DriftDetectionSystem [1]
centralprocessor: CentralProcessor [1]
communicationsystem: CommunicationSystem [1]
obstacledetectionsystem: ObstacleDetectionSystem [1]
wheel: WheelController [1]

CONTENT

- Introduction
- **Basic Elements**
- Port
- Connector
- Quiz
- Summary

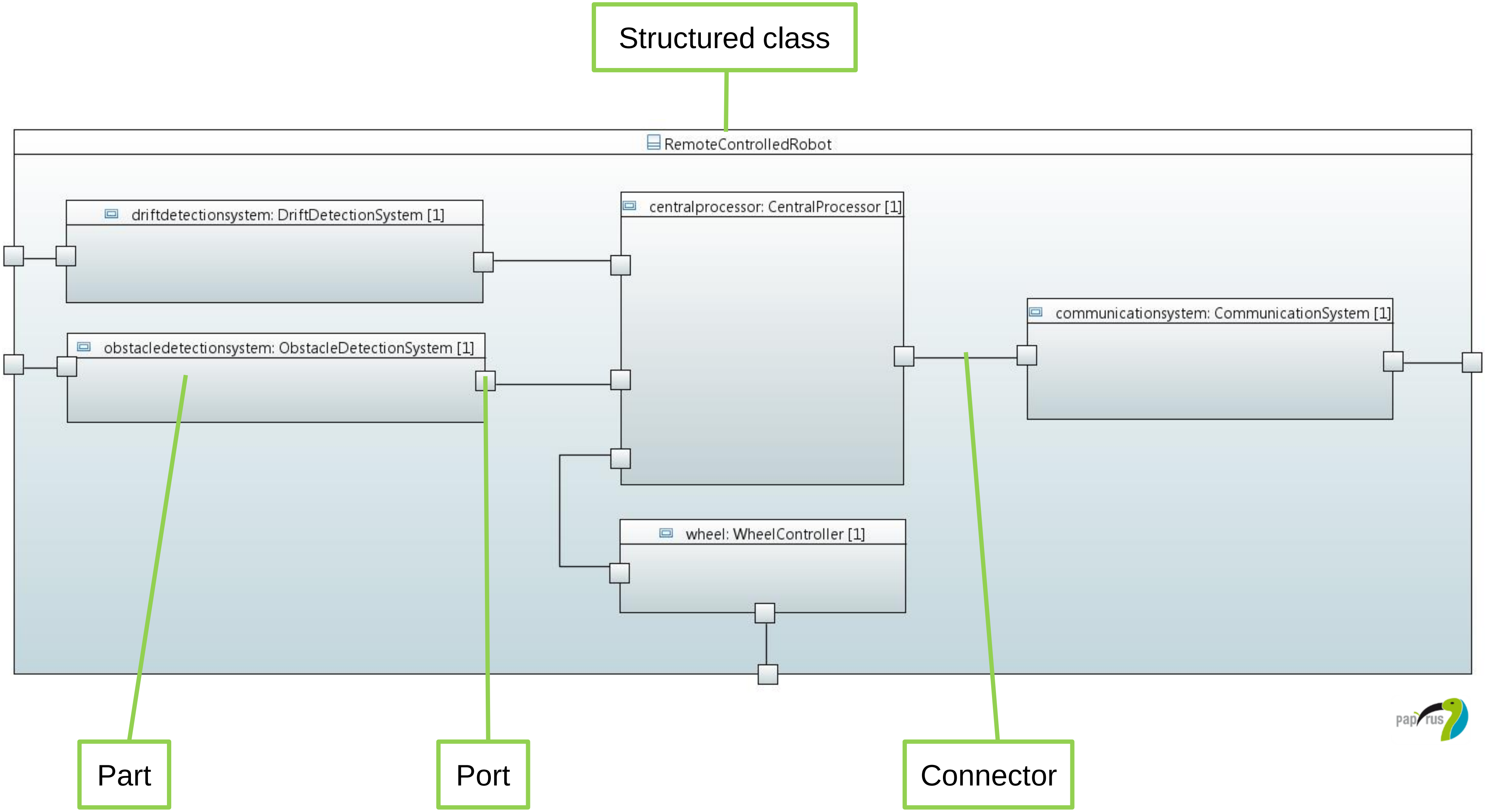
Key concepts of a composite structure

- Part: role played by some instance (or collection of instances) of a class; modeled with a property owned by the composite

ⓘ Remember that an association end is a role and may be an attribute of the class C at the opposite end, i.e. modeled with a property owned by C!

- Port: an interaction point between a class and its environment, or its internal parts
- Connector: shows possible interaction by binding elements, who may be parts of ports
- Structured class: the class meta-element, introduced previously, inherits structured classifier. This means a class may own ports and parts, and its behavior can be described completely or partially through interaction between its parts

Example of a composite structure


Structured class
RemoteControlledRobot
driftdetectionsystem: DriftDetectionSystem [1]
centralprocessor: CentralProcessor [1]
communicationsystem: CommunicationSystem [1]
obstacledetectionsystem: ObstacleDetectionSystem [1]
wheel: WheelController [1]
papyrus
Part
Port
Connector

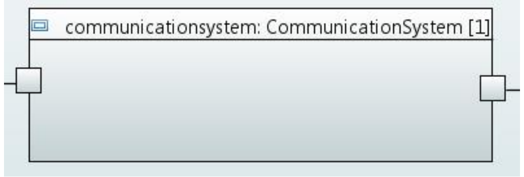
CONTENT

- Introduction
- Basic Elements
- **Port**
- Connector
- Quiz
- Summary

PORT

Overview

- Interaction point between a class and its environment, or its internal parts
- Owned by a class
- Public visibility by default
- A port can have a multiplicity if there are several ports with the same specification
- Two kinds: service port and behavior port (not mutual exclusive)
- Shown in a composite structure diagram on the border of some part:

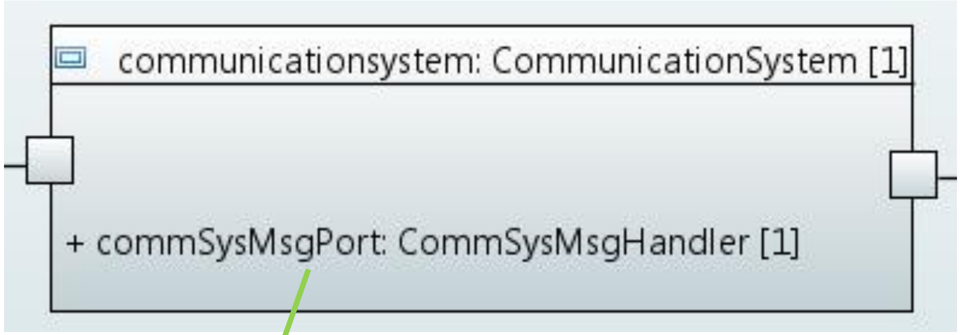


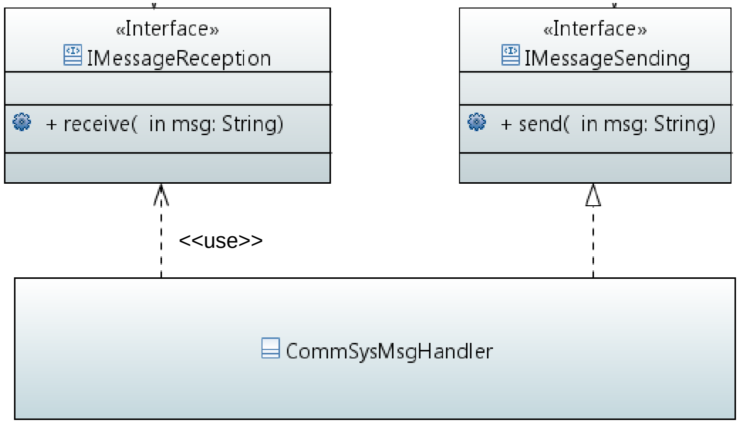
ⓘ Careful with this graphical notation: it may seem as the port is owned by the property representing the part, but it is actually owned by the class that types the property. Ports owned by a property have been discussed, to answer a need by some industrial systems, but not such thing exists yet.

Service port

- **Denotes a class provides/requires services to/from its environment:**
  - Provided interfaces: list of interfaces provided by the class through this port, i.e. requests the environment may make to this class (e.g. operations that can be called through this port)
  - Required interfaces: list of interfaces required by the class through this port, i.e. requests this port may make to the environment (e.g. operations that this class can call through this port)
- **Type of a port:**
  - Specifies the provided/required interfaces of a port:
    - If the type is a class, the port provides/requires interfaces realized/used by the class
    - If the type is an interface, the port provides the interface
  - You can also type a port with a class to specify more sophisticated behavior when requests arrive at this port

Service port example

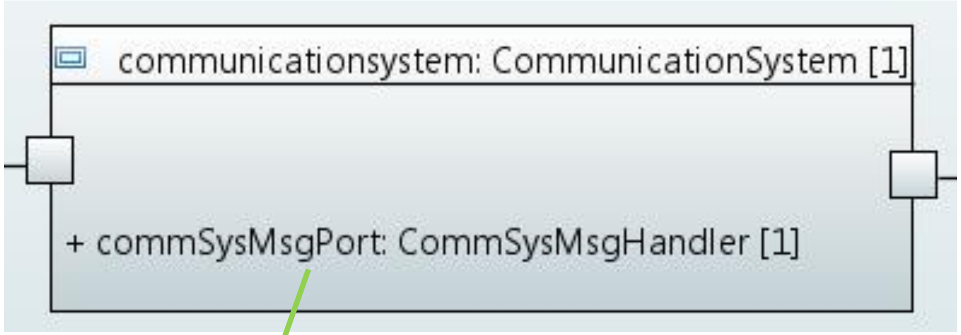

communicationsystem: CommunicationSystem [1]
+ commSysMsgPort: CommSysMsgHandler [1]

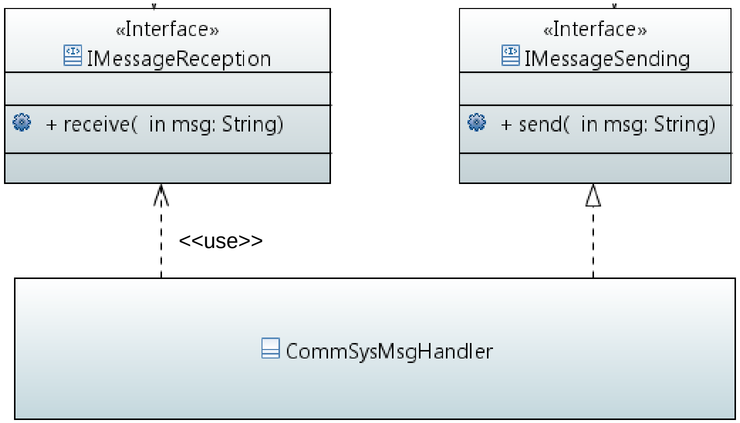


Port “commSysMsgPort” is typed by class “CommSysMsgHandler” which realizes “IMessageSending” and uses “IMessageReception”

What interface(s) does commSysMsgPort provides and requires?

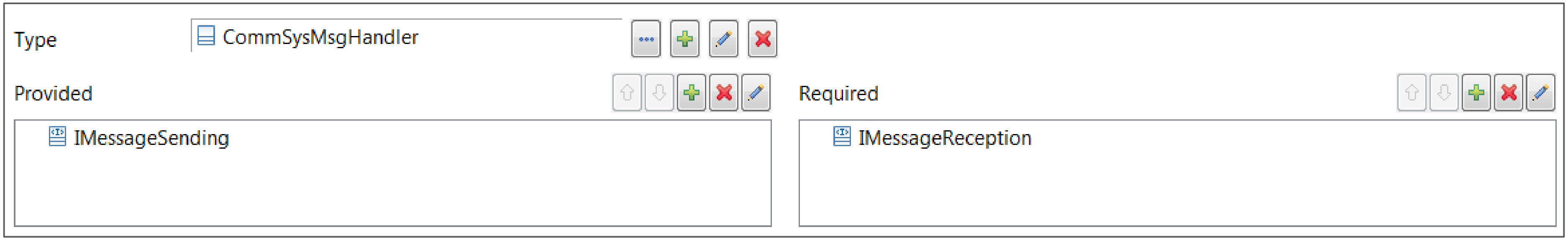
Service port example


communicationsystem: CommunicationSystem [1]
+ commSysMsgPort: CommSysMsgHandler [1]



Port “commSysMsgPort” is typed by class “CommSysMsgHandler” which realizes “IMessageSending” and uses “IMessageReception”

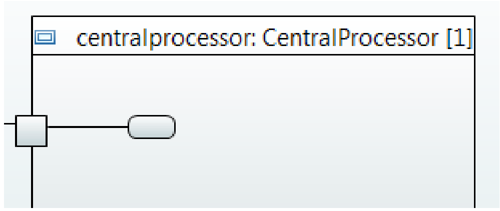
Provides IMessageSending and requires IMessageReception



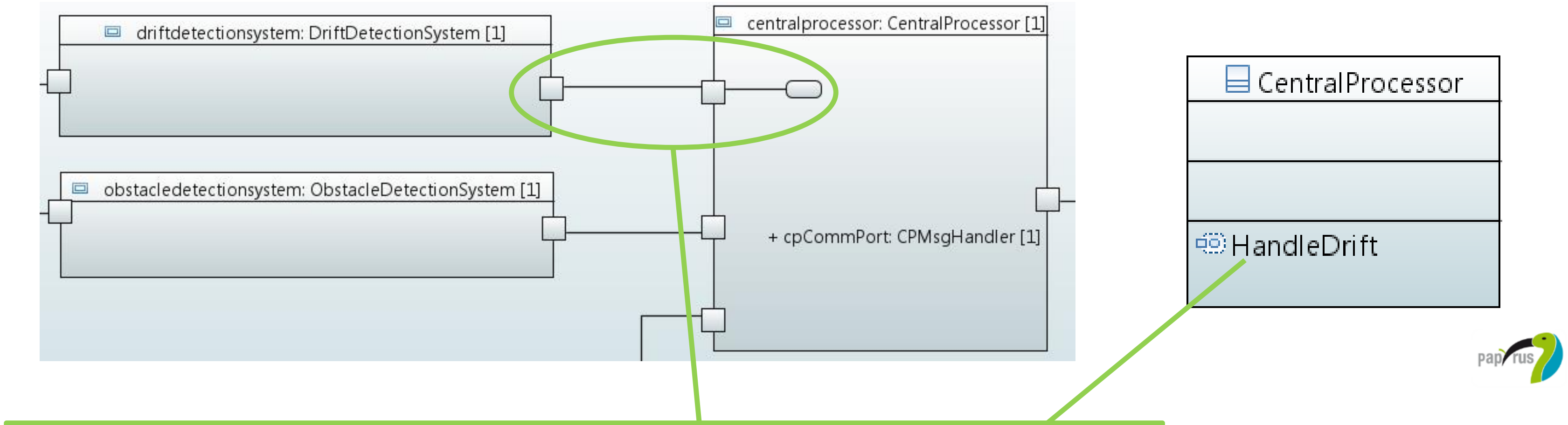
Type	CommSysMsgHandler
Provided	IMessageSending
Required	IMessageReception

Behavior port

- By default ports are not behavior ports: usually a port forwards incoming requests to some a part of the class
- If the class itself handles the request, without the need of some internal part, then the port is connected to a behavior of the class
- The port is then a behavior port
- If no behavior to handle the request is specified, the request is lost
- Shown as a port connected to some state symbol



Behavior port example


driftdetectionsystem: DriftDetectionSystem [1]
obstacledetectionsystem: ObstacleDetectionSystem [1]
centralprocessor: CentralProcessor [1]
+ cpCommPort: CPMsgHandler [1]
CentralProcessor
HandleDrift
papyrus

“Central processor” is connected to “drift detection system” through a behavior port. Incoming drift detections will be handled by the “HandleDrift” behavior owned by the “CentralProcessor” class

CONTENT

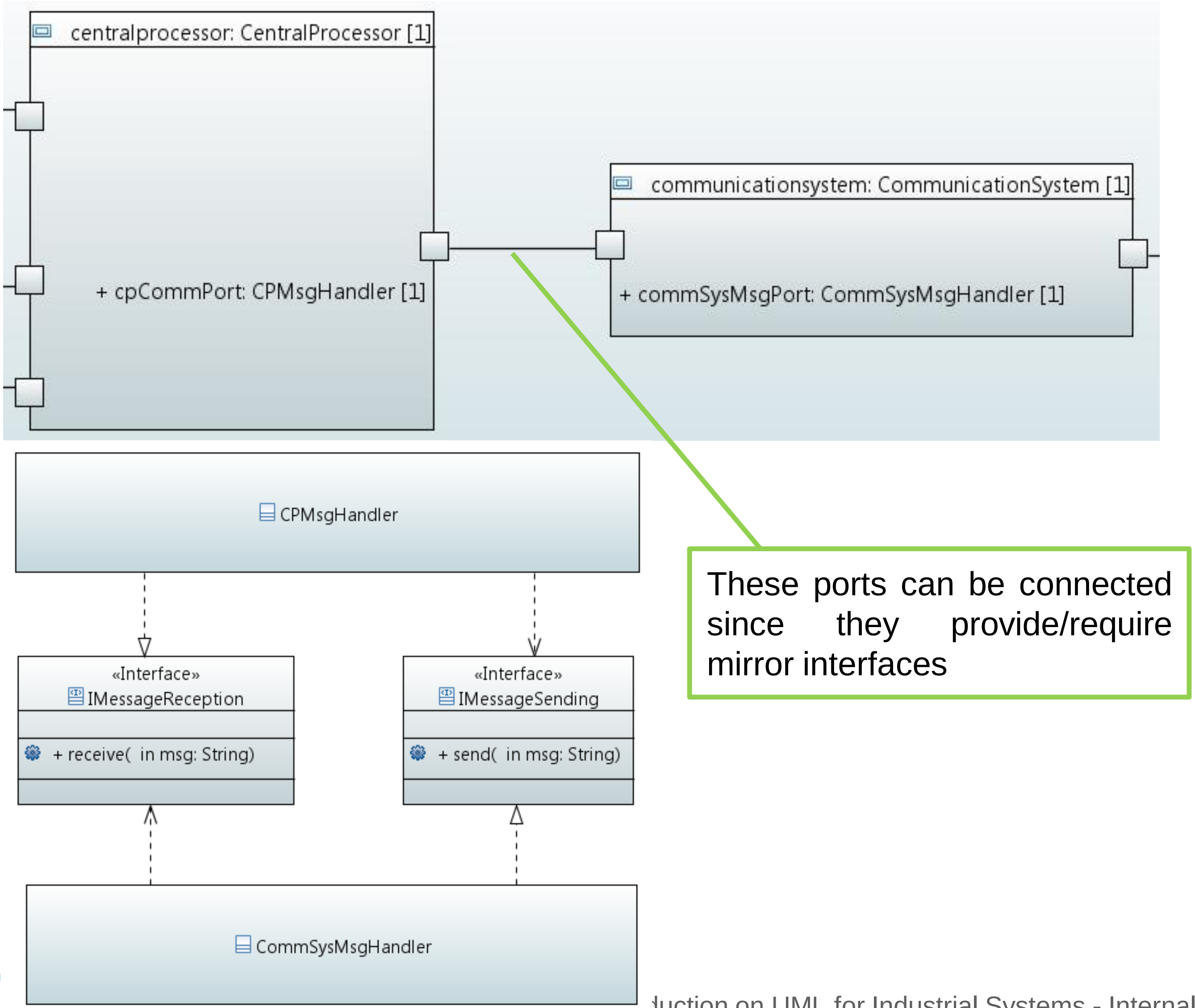
- Introduction
- Basic Elements
- Port
- **Connector**
- Quiz
- Summary

CONNECTOR

Overview

- Binds two parts or ports to show possible interaction between the two
- Can represent an instance of an association, i.e. can be typed by an association, but it is not mandatory
- For parts, if part P1 is typed by class C1 who uses some interface(s), then part P2 at the other end must be typed by class C2 who realizes the same interface(s)
- For ports, if port P1 at one end requires some interface(s), then port P2 at the other end must provide the same interface(s)

Example


centralprocessor: CentralProcessor [1]
+ cpCommPort: CPMsgHandler [1]
communicationsystem: CommunicationSystem [1]
+ commSysMsgPort: CommSysMsgHandler [1]
CPMsgHandler
«Interface»
IMessageReception
+ receive(in msg: String)
«Interface»
IMessageSending
+ send(in msg: String)
CommSysMsgHandler

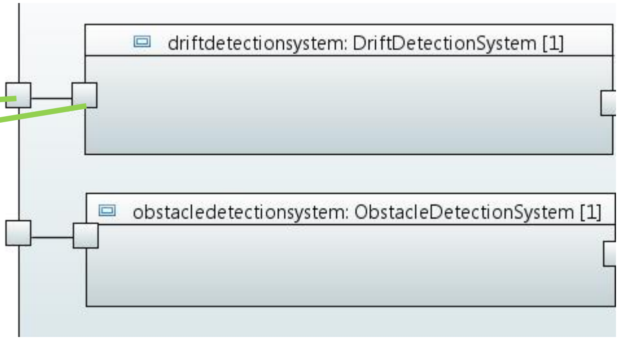
These ports can be connected since they provide/require mirror interfaces

Delegation

- A port P1 that provides some interface(s) can be connected to a port P2 that provides the same interface(s)
- Note that this does not contract the rule on how ports requiring interface(s) should be connected

? When do we typically connect two ports that both provide the same interface(s)?

! When we delegate some incoming requests to an internal part


driftdetectionsystem: DriftDetectionSystem [1]
obstacledetectionsystem: ObstacleDetectionSystem [1]

Both of these ports provide the same interface(s)

CONTENT

- Introduction
- Basic Elements
- Port
- Connector
- **Quiz**
- Summary

QUIZ

?

What are the basic elements of a composite structure model?

!

Part (property), port, connector, structured class

QUIZ

?

What is a service port? How is it specified?

!

A service port denotes a class provides/requires services (modeled with interfaces) to/from its environment. Typing the port specifies the list of provided/required interfaces.

QUIZ

?

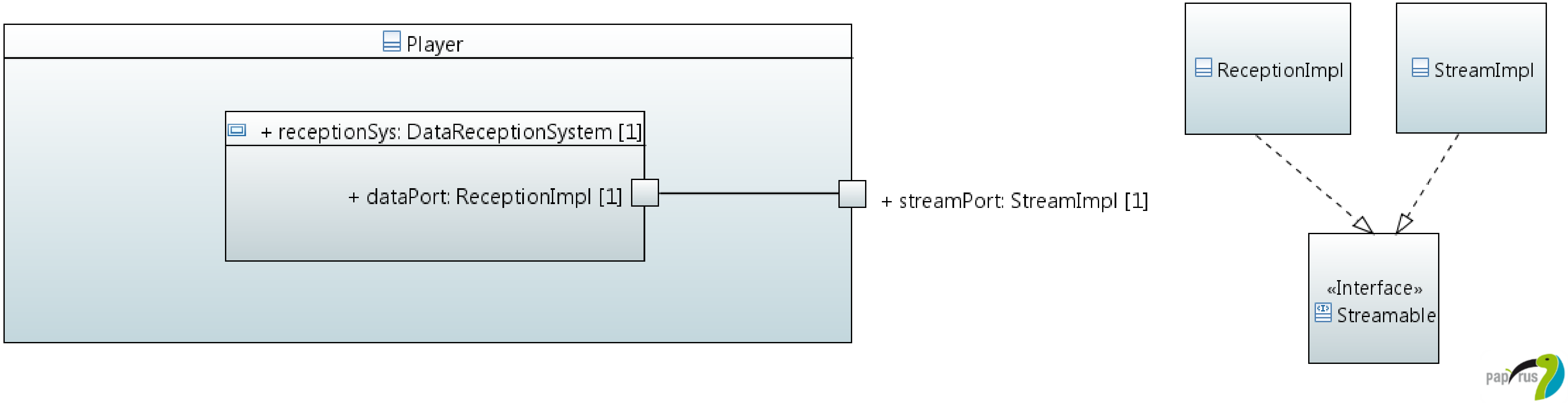
What is a behavior port? What happens when a request arrives at a behavior port and there is not specified behavior to handle the request?

!

A behavior port is connected to a behavior of a class, rather than some internal part. The request is lost if there is not specified behavior to handle it.

?

What’s wrong here?


Player
+ receptionSys: DataReceptionSystem [1]
+ dataPort: ReceptionImpl [1]
+ streamPort: StreamImpl [1]
ReceptionImpl
StreamImpl
«Interface»
Streamable
papyrus

!

Nothing! The streamPort is a typical a delegation port since it provides the same interface as dataPort.

CONTENT

Introduction

Basic Elements

Port

Connector

Quiz

Summary

SUMMARY

- Modeling the internal structure of a class lets us have a better view of the collaboration between the parts of the class
- We describe a composite structure in UML
- In a composite structure, parts interact through ports owned by the class typing the part
- Ports are connected by connectors that show a possible interaction
- A port is an interaction point that provides/requires some interface(s), which are the services provided/required by the class
- The provided/required interfaces of a port must be conform to the type of the port, which may be an interface or a class that realizes/uses interfaces
- The port at one end of a connector must provide the interfaces required by the port at the other end
- A port may be a behavior port, connected to a behavior of the class, rather than some internal part of the class; the behavior handles directly incoming requests at the port