SRML

wires and interaction protocols

Laura Bocchi
bocchi@mcs.le.ac.uk

Agenda

- Wires
- Interaction Protocols

Wires

BUSINESS PROTOCOL Warehouse **is**

INTERACTIONS

r&s tellShipAvail

🔔 which:product, many:nat

snd makePayment

rcv shipOrder

BEHAVIOUR

...

BUSINESS PROTOCOL Warehouse **is**

INTERACTIONS

r&s checkShipAvail

🔔 which:product, many:nat

snd makePayment

rcv shipOrder

BEHAVIOUR

...

BUSINESS ROLE Broker **is**

INTERACTIONS

r&s requestQuote

🔔 which:product

✉ cost:money

r&s orderGoods

🔔 many:nat

✉ much:money

s&r checkShipAvail

🔔 which:product, many:nat

rcv makePayment

snd shipOrder

rcv confirmShip

ask how(product):money

...

ORCHESTRATION

...

- Often we assumed, for simplicity, that the names for the interactions and the parameters are pairwise corresponding
- In general, we can reuse specifications. This could cause mismatching of some names or even duplication (if two nodes have the same specification)

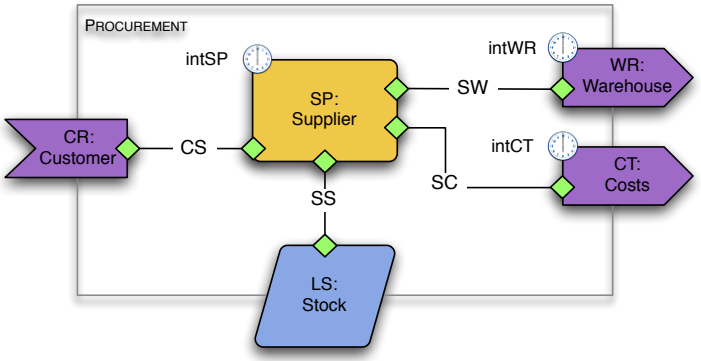
Wires and connectors

- Wires specify the correspondence between interaction/parameter names of different nodes
- E.g., SW specifies the correspondence between SP and WA
- A wire is defined as one or more connectors

SP Supplier	c_4	**SW**	d_4	**WA** Warehouse
s&r checkShipAvail which many	S_1 i_1 i_2	Straight	R_1 o_1 oi_2	**r&s** tellShipAvail which many
rcv confirmShip	S_1	Straight	R_1	**snd** confirmShip

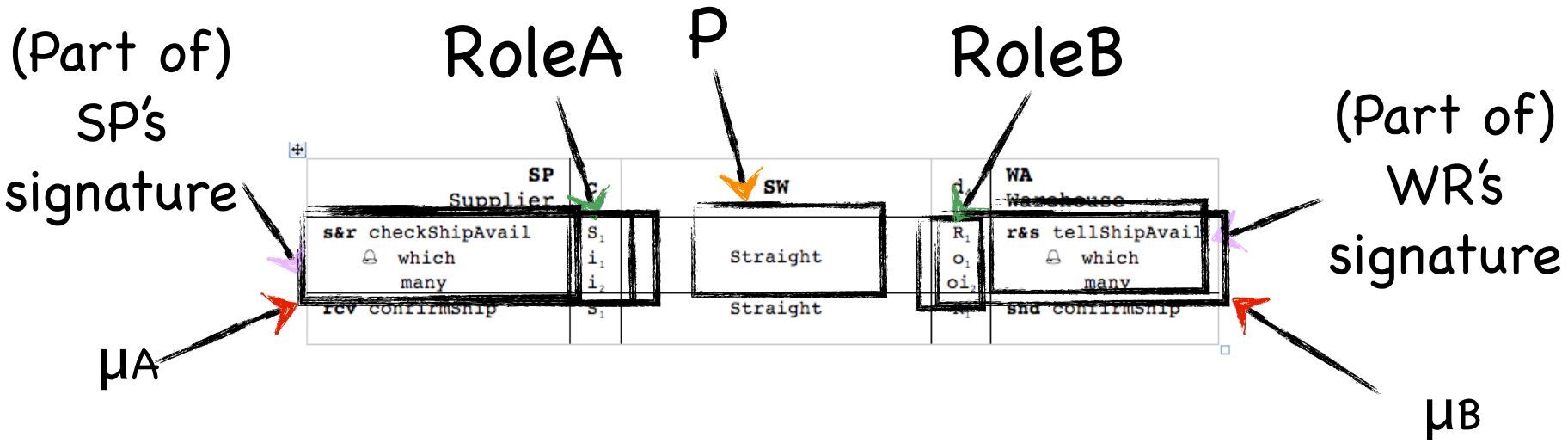
each line represents a connector

wire



Wires and connectors

- A connector is a triple: < μ_A , P , μ_B > where
  - P is an interaction protocol. We use $role A_P$ and $role B_P$ to designate its roles and $glue_P$ to designate the coordination
  - μ_A and μ_B are attachments that connect the roles of P to the signatures of the connected nodes



A simple interaction protocol

INTERACTION PROTOCOL Straight is

ROLE A
  s&r S_1
    🔔 i_1**:product**
      i_2**:money**
ROLE B
  r&s R_1
    🔔 o_1**:product**
      o_2**:money**
COORDINATION
$S_1 \equiv R_1$
$S_1.i_1=R_1.o_1$
$S_1.i_2=R_1.o_2$

SP Supplier	c_4	**SW**	d_4	**WA** Warehouse
s&r checkShipAvail 🔔 which many	S_1 i_1 i_2	Straight	R_1 o_1 oi_2	**r&s** tellShipAvail 🔔 which many
rcv confirmShip	S_1	Straight	R_1	**snd** confirmShip

the events associated to S1 are the same as the events associated to R1

a one-to-one correspondence is stated between parameters

Another interaction protocol

INTERACTION PROTOCOL RobinHood is

ROLE A
s&r S_1
🔔 **i_1:product**
✉ **o_1:money**
ROLE B
r&s R_1
🔔 **i_1:product**
✉ **o_1:money**
COORDINATION
$S_1 \equiv R_1$
$S_1.i_1 = R_1.i_1$
$S_1.o_1 > 100 \supset S_1.o_1 = R_1.o_1 - 50$

the data can be also elaborated

if the amount is > 100£ then the protocol RobinHood steals 50£

A simple interaction protocol

INTERACTION PROTOCOL Straight is

ROLE A
s&r S_1
🔔 **i_1:product**
✉ **o_1:money**
ROLE B
r&s R_1
🔔 **i_1:product**
✉ **o_1:money**
COORDINATION
$S_1 \equiv R_1$
$S_1.i_1=R_1.i_1$
$S_1.o_1=R_1.o_1$

INTERACTION PROTOCOL Straight is

ROLE A
s&r S_1
🔔 **i_1:product**
i_2:usrId
✉ **o_1:money**
✔ **c_1:payData**
ROLE B
r&s R_1
🔔 **i_1:product**
i_2:usrId
✉ **o_1:money**
✔ **c_1:payData**
COORDINATION
$S_1 \equiv R_1$
$S_1.i_1=R_1.i_1$
$S_1.i_2=R_1.i_2$
$S_1.o_1=R_1.o_1$
$S_1.c_1=R_1.c_1$

- Straight can be used only on a couple of conversational interactions (s&r and r&s) that have exactly one 🔔-parameter and exactly one ✉-parameter
- For couples of conversational interaction with a different number of parameters we must define another interaction protocol (e.g., two 🔔-parameter, one ✉-parameter and one ✔-parameters)

A simple interaction protocol

INTERACTION PROTOCOL Straight is

ROLE A
s&r S_1
🔔 **i_1:product**
i_2:usrId
✉ **o_1:money**
✔ **c_1:payData**
ROLE B
r&s R_1
🔔 **i_1:product**
i_2:usrId
✉ **o_1:money**
✔ **c_1:payData**
COORDINATION
$S_1 \equiv R_1$
$S_1.i_1=R_1.i_1$
$S_1.i_2=R_1.i_2$
$S_1.o_1=R_1.o_1$
$S_1.c_1=R_1.c_1$

INTERACTION PROTOCOL Straight is

ROLE A
s&r S_1
🔔 **i_1:destination**
i_2:outdate
✉ **o_1:money**
✔ **c_1:payData**
ROLE B
r&s R_1
🔔 **i_1:destination**
i_2:outdate
✉ **o_1:money**
✔ **c_1:payData**
COORDINATION
$S_1 \equiv R_1$
$S_1.i_1=R_1.i_1$
$S_1.i_2=R_1.i_2$
$S_1.o_1=R_1.o_1$
$S_1.c_1=R_1.c_1$

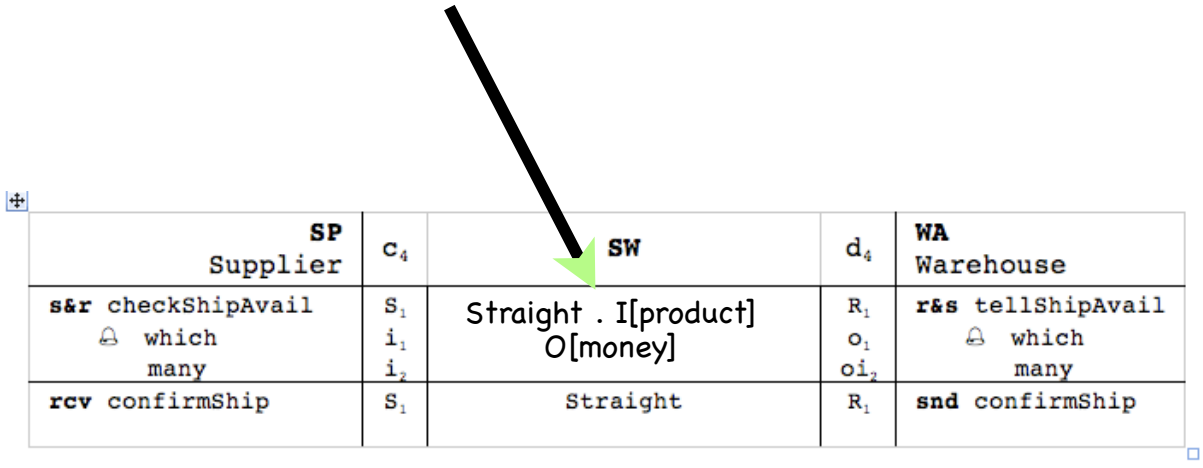
- The protocol above can be used only if the parameters are all of type product, usrId, money and payData
- If we want to use, say, destination, outdate, money and paydata we have to define another protocol
- Otherwise we can parametrize interaction protocols …

A parametrized interaction protocol

- Straight.I(d1)O(d2) can be used only on a couple of conversational interactions (s&r and r&s) that have exactly one 🔔-parameter and exactly one ✉-parameter...
- But it can be used also for interaction that carry other data types than product and money

INTERACTION PROTOCOL Straight.I(d_1)O(d_2) is

ROLE A
  s&r S_1
    🔔 i_1:d_1
       i_2:d_2
ROLE B
  r&s R_1
    🔔 o_1:d_1
       o_2:d_2
COORDINATION
$S_1 \equiv R_1$
$S_1.i_1 = R_1.o_1$
$S_1.i_2 = R_1.o_2$



SP Supplier	c_4	**SW**	d_4	**WA** Warehouse
s&r checkShipAvail 🔔 which many	S_1 i_1 i_2	Straight . I[product] O[money]	R_1 o_1 oi_2	**r&s** tellShipAvail 🔔 which many
rcv confirmShip	S_1	Straight	R_1	**snd** confirmShip

Example: parameters mismatching

INTERACTION PROTOCOL Internal2SMS **is**

ROLE A

snd S_1

i_1:phoneNum
i_2:reference
i_3:string
i_4:geoData

ROLE B

rcv R_1

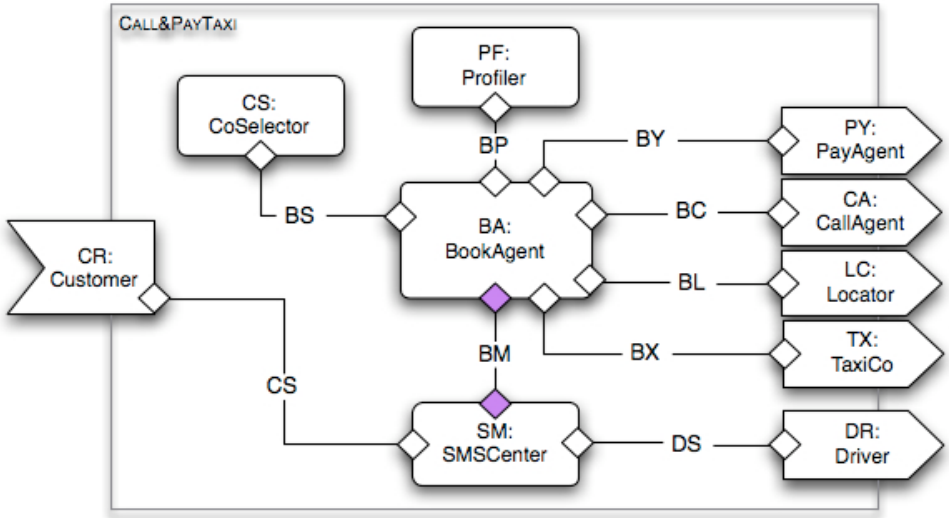
i_1:phoneNum
i_2:string

LOCAL

textify:reference,string,geoData→string

COORDINATION

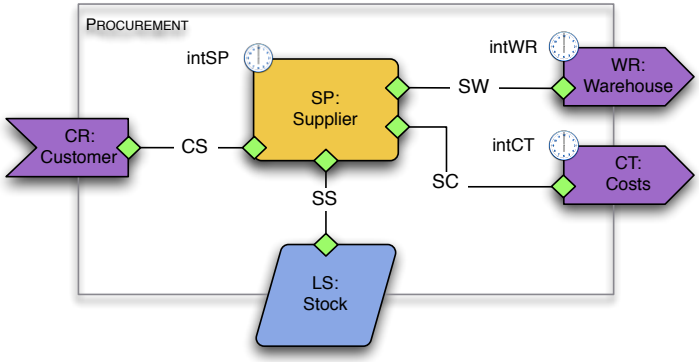
$S_1 \equiv R_1$
$S_1.i_1 = R_1.i_1$
$R_1.i_2 = \text{textify}(S_1.i_2, S_1.i_3, S_1.i_4)$



BA BookAgent		**BM**		**SM** SMSCentre(4777)
snd informCustomer	S_1		R_1	**rcv** forwardOUT[1]
phone	i_1		i_1	destination
taxiNum	i_2	**Internal2SMS**	i_2	text
callCode	i_3			
location	i_4			

Example: add authentication

- What if we wish to add authenitcation in the interactions between WR and SP?
- Well, we could modify Supplier and Warehouse
- But what if we decide to use services for warehouses with authentication but we do not want to change Supplier?
- We can add authentication in the interaction protocol

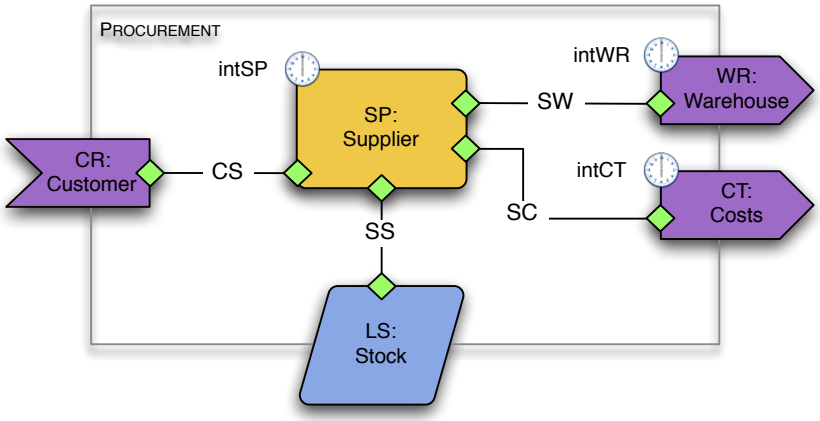


SP Supplier	c_4	**SW**	d_4	**WA** Warehouse
s&r checkShipAvail 🔔 which many	S_1 i_1 i_2	Secure	R_1 o_1 oi_2	**r&s** tellShipAvail 🔔 which many
rcv confirmShip	S_1	Straight	R_1	**snd** confirmShip

INTERACTION PROTOCOL Secure is

ROLE A
s&r S_1
🔔 **i_1:product**
✉ **o_1:money**
ROLE B
r&s R_1
🔔 **i_1:product**
i_2:password
✉ **o_1:money**
COORDINATION
$S_1 \equiv R_1$
$S_1.i_1=R_1.i_1$
$S_1.o_1=R_1.o_1$
$R_1.i_2$="secret"

Let's go back to the example...



- EX-Ps describe what the module provides
- EX-Rs describe what the module requires
- EX-Is are not a node that executes but just a description
- In fact, what CR "provides" is what SP "provides"
- In fact, WR describes what some other node in another module "provides" and we have to interact to

The protocol of the EX-P

```
BUSINESS PROTOCOL Customer is
  INTERACTIONS
   r&s requestQuote
    🔔  which:product
    ✉  cost:money
   r&s orderGoods
    🔔  many:nat
    ✉  much:money
   rcv makePayment
   snd shipOrder
  BEHAVIOUR
   ...
```

EX-P have the same "direction" of the node to which they are connected

```
BUSINESS ROLE Supplier is
  INTERACTIONS
   r&s requestQuote
    🔔  which:product
    ✉  cost:money
   r&s orderGoods
    🔔  many:nat
    ✉  much:money
   rcv makePayment
   snd shipOrder
   s&r checkShipAvail
    🔔 which:product, many:nat
   rcv confirmShip
   ask how(product):money
   ask checkStock(product,nat):bool
   tll incStock(product,nat)
   tll decStock(product,nat)
  BEHAVIOUR
   ...
```

The protocol of the EX-R

```
BUSINESS ROLE Supplier is
  INTERACTIONS
   r&s requestQuote
    🔔  which:product
    ✉  cost:money
   r&s orderGoods
    🔔  many:nat
    ✉  much:money
   rcv makePayment
   snd shipOrder
   s&r checkShipAvail
    🔔 which:product, many:nat
   rcv confirmShip
   ask how(product):money
   ask checkStock(product,nat):bool
   tll incStock(product,nat)
   tll decStock(product,nat)
  ORCHESTRATION
   ...
```

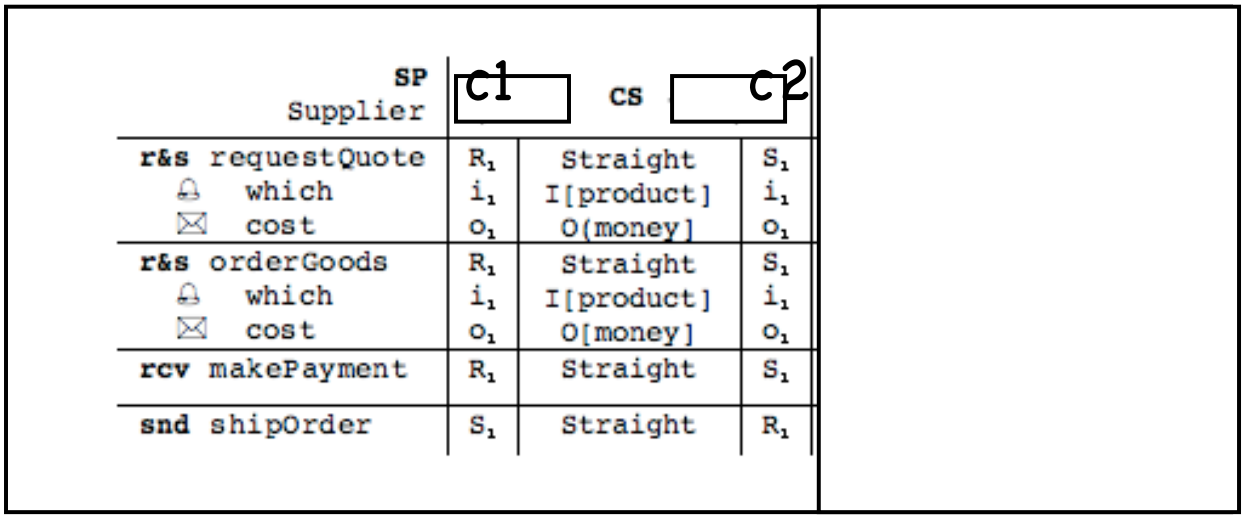
```
BUSINESS PROTOCOL Warehouse is
  INTERACTIONS
   r&s checkShipAvail
    🔔  which:product, many:nat
   snd confirmShip
   BEHAVIOUR
   ...
```

EX-R have complementary "direction" with respect to the node to which they are connected

The wires to EX-I

SP Supplier	c_4	**SW**	d_4	**WA** Warehouse
s&r checkShipAvail 🔔 which many	S_1 i_1 i_2	Straight	R_1 o_1 oi_2	**r&s** tellShipAvail 🔔 which many
rcv confirmShip	S_1	Straight	R_1	**snd** confirmShip

Notice that SP (component) and WA (EX-R) have complementary interaction types



The specification of the wires that connect module components to the provides-interface use a slightly different syntax.