Interaction safety in Web service orchestrations

Just sessionize it !

Jonathan Michaux
6/11/2012

TELECOM ParisTech

Département INFRES

I am going to present to you:

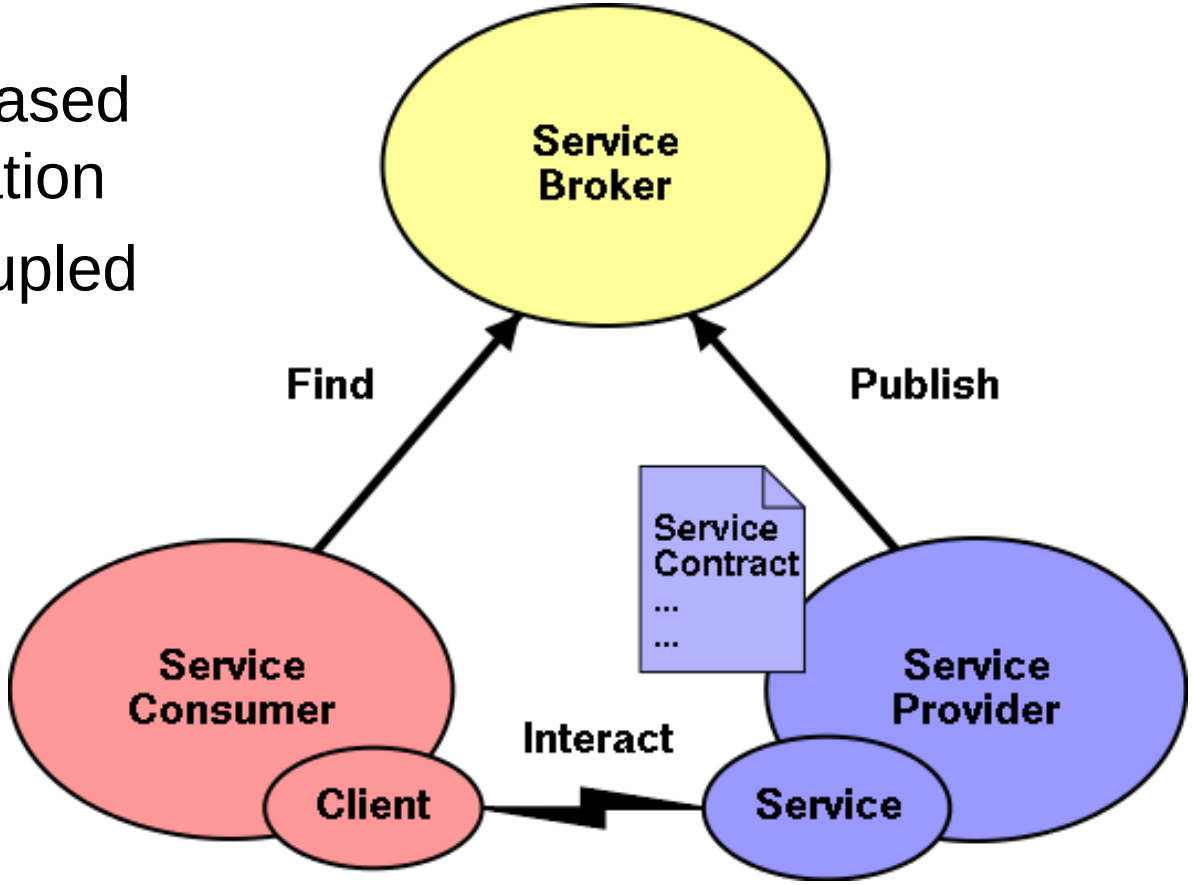
- The meaning and purpose of Web service orchestrations
- Runtime errors that we would like to be able to avoid
- A simple orchestration language
- A new approach to behavioural typing: session-based programming
- **How the approach helps us avoid the runtime errors *at compile-time***

To do so, I am going to:

- Introduce Web services, orchestration, and interaction errors
- Present a simple orchestration language with a promising new feature: **typed sessions** (a concurrent programming paradigm with lots of potential)
- Jump into an example of an orchestration
  - Implement the orchestration
  - Show that the implementation contains errors
  - Show how the errors are detected
  - Correct the code

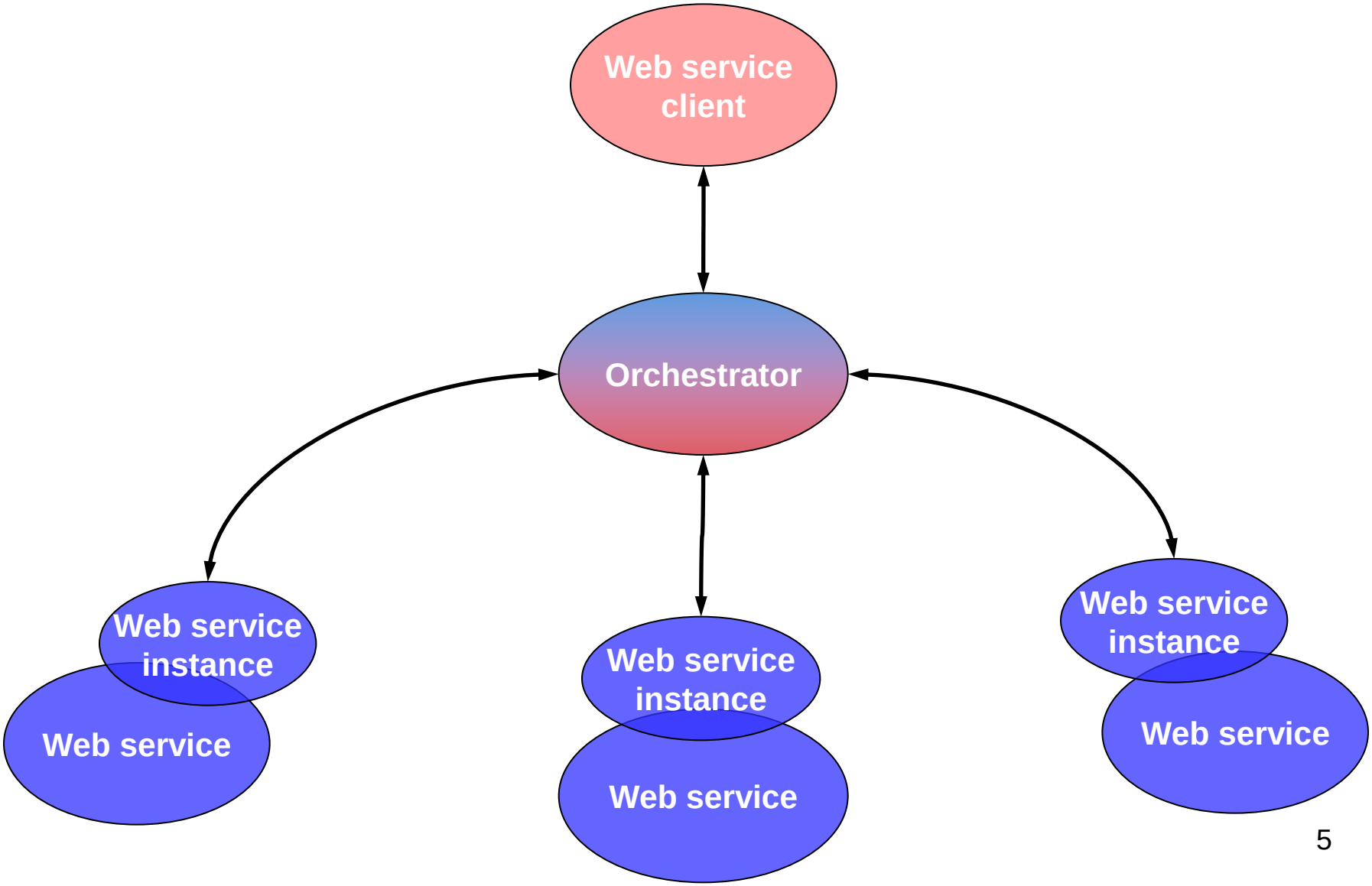
Service Oriented Architecture (SOA)

- Message-based communication
- Loosely coupled services



(w3c.org)

Web service orchestration

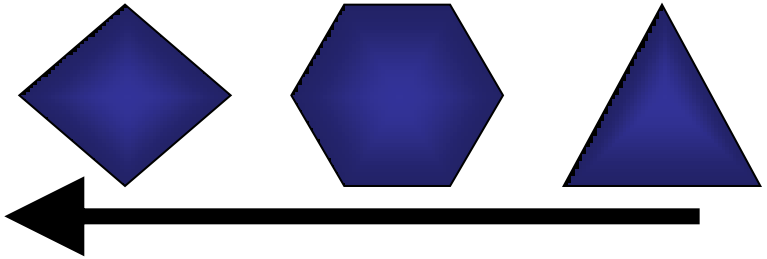


Interactions errors

A service client is awaiting message

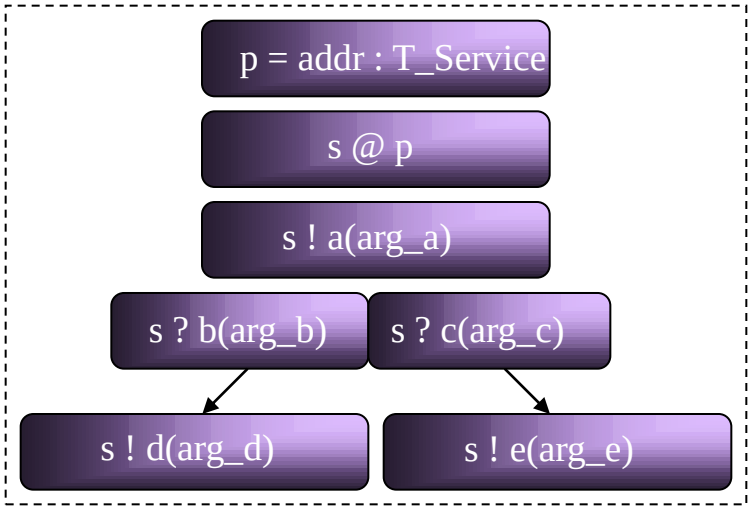
or

… but the message queue contains



A simple orchestration language with typed sessions

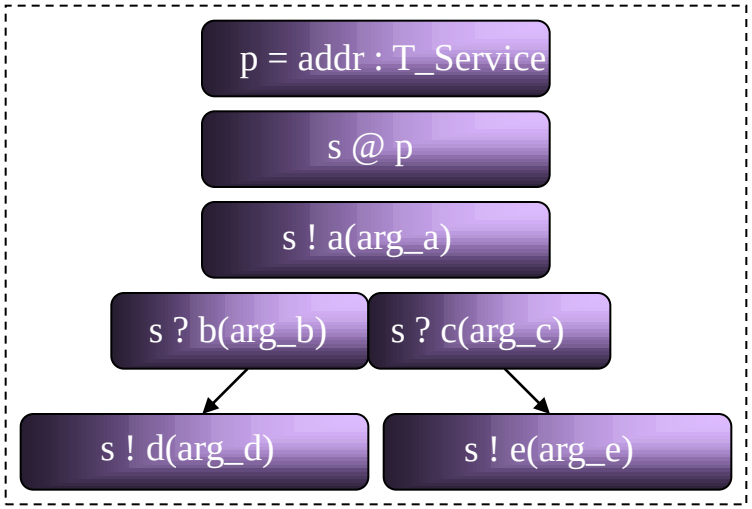
- **Atomic activities: Invocation, Reception, Session initiation**
- **Structured activities: Sequence, Parallel, Pick, Recursion…**
- **Control links: Synchronisation of parallel activities.**



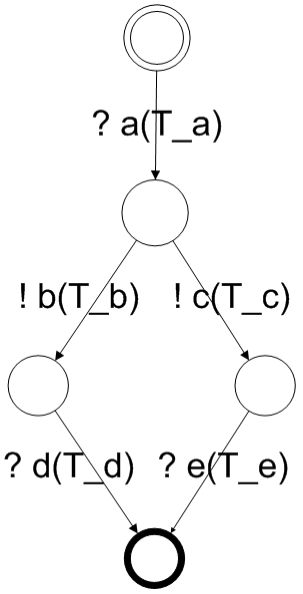
Traditional **computation** tasks can be done externally.

A simple orchestration language with typed sessions

- **Atomic activities: Invocation, Reception, Session initiation**
- **Structured activities: Sequence, Parallel, Pick, Recursion…**
- **Control links: Synchronisation of parallel activities.**
- **Session types: Behavioural types that specify allowed interactions.**

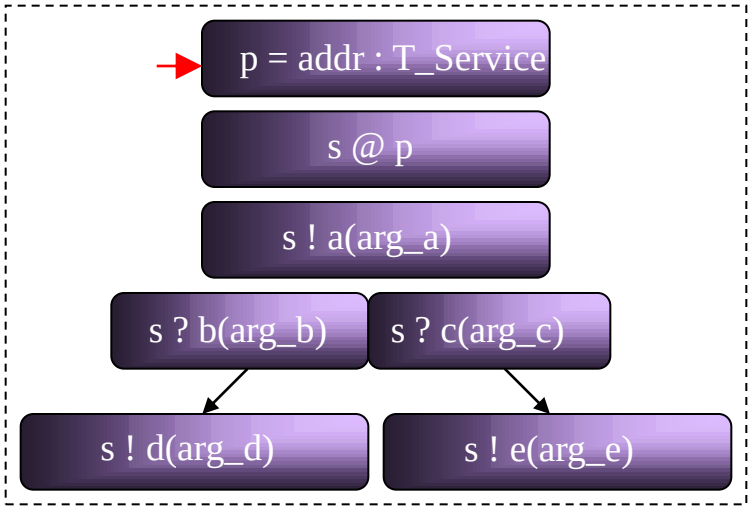


T_Service service type:

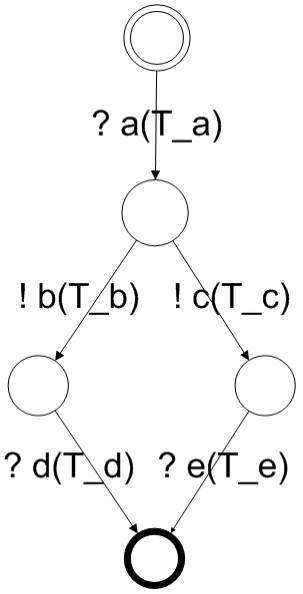


A simple orchestration language with typed sessions

- **Atomic activities: Invocation, Reception, Session initiation**
- **Structured activities: Sequence, Parallel, Pick, Recursion…**
- **Control links: Synchronisation of parallel activities.**
- **Session types: Behavioural types that specify allowed interactions.**

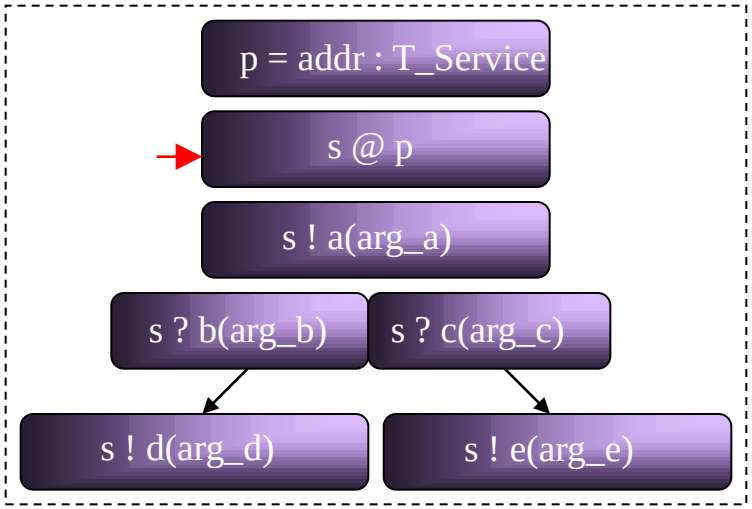


T_Service service type:

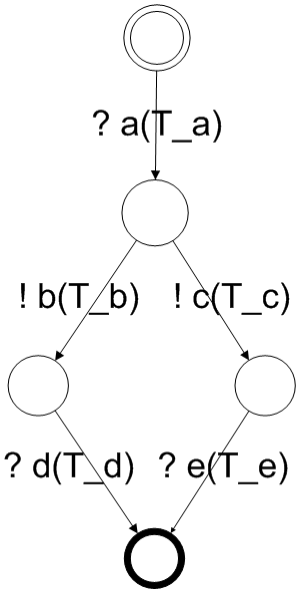


A simple orchestration language with typed sessions

- **Atomic activities: Invocation, Reception, Session initiation**
- **Structured activities: Sequence, Parallel, Pick, Recursion…**
- **Control links: Synchronisation of parallel activities.**
- **Session types: Behavioural types that specify allowed interactions.**

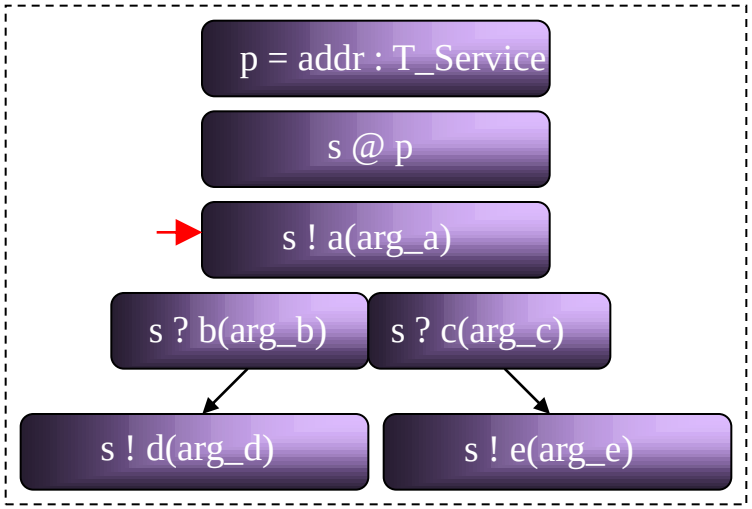


T_Service service type:

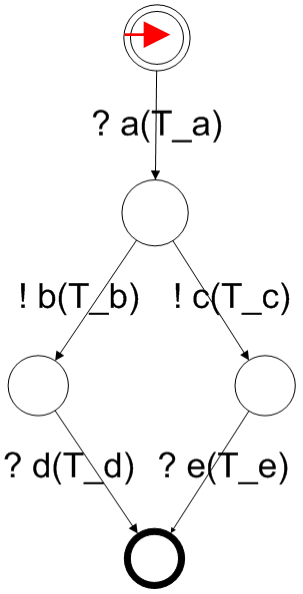


A simple orchestration language with typed sessions

- **Atomic activities: Invocation, Reception, Session initiation**
- **Structured activities: Sequence, Parallel, Pick, Recursion…**
- **Control links: Synchronisation of parallel activities.**
- **Session types: Behavioural types that specify allowed interactions.**

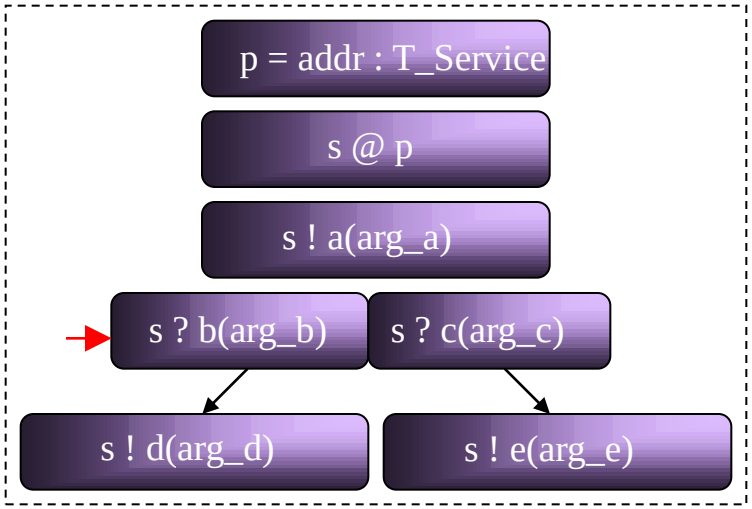


T_Service service type:

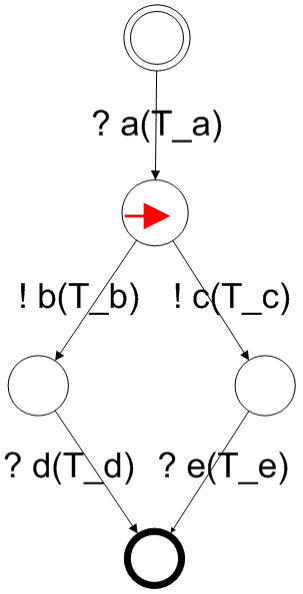


A simple orchestration language with typed sessions

- **Atomic activities: Invocation, Reception, Session initiation**
- **Structured activities: Sequence, Parallel, Pick, Recursion…**
- **Control links: Synchronisation of parallel activities.**
- **Session types: Behavioural types that specify allowed interactions.**

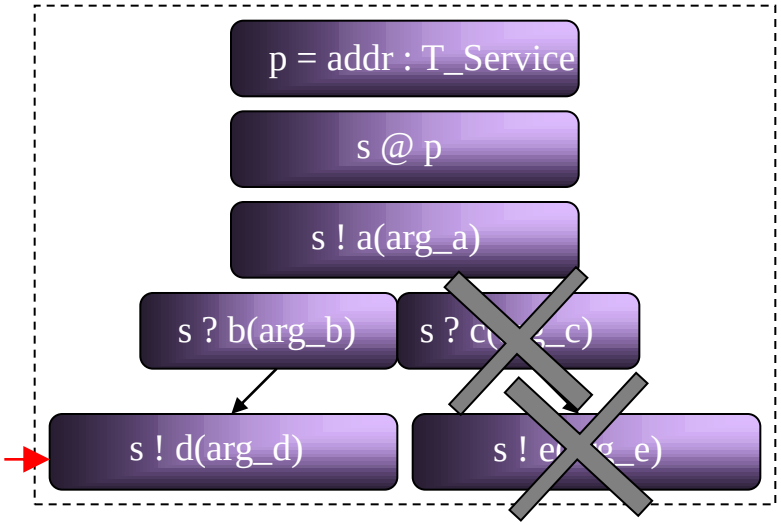


T_Service service type:

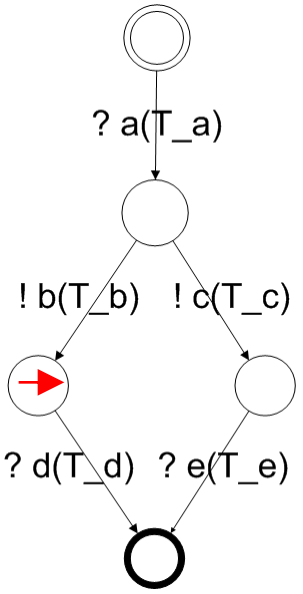


A simple orchestration language with typed sessions

- **Atomic activities: Invocation, Reception, Session initiation**
- **Structured activities: Sequence, Parallel, Pick, Recursion…**
- **Control links: Synchronisation of parallel activities.**
- **Session types: Behavioural types that specify allowed interactions.**

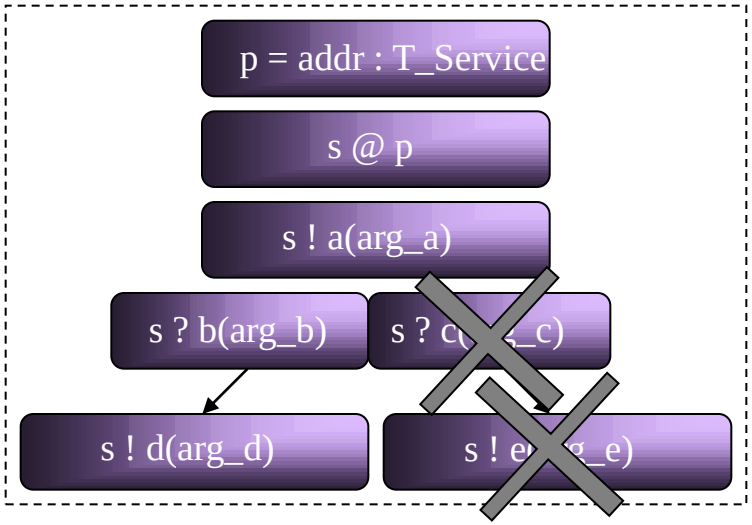


T_Service service type:

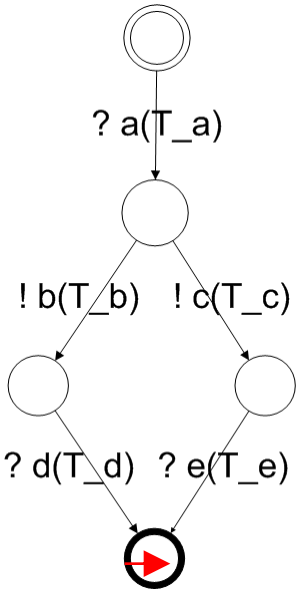


A simple orchestration language with typed sessions

- **Atomic activities: Invocation, Reception, Session initiation**
- **Structured activities: Sequence, Parallel, Pick, Recursion…**
- **Control links: Synchronisation of parallel activities.**
- **Session types: Behavioural types that specify allowed interactions.**



T_Service service type:



Example service orchestration: The Event Booking System

The Event Booking System chooses a date from a previously set up Doodle page and books a room by using EOLE and some equipment by using the DSI service.

Doodle®

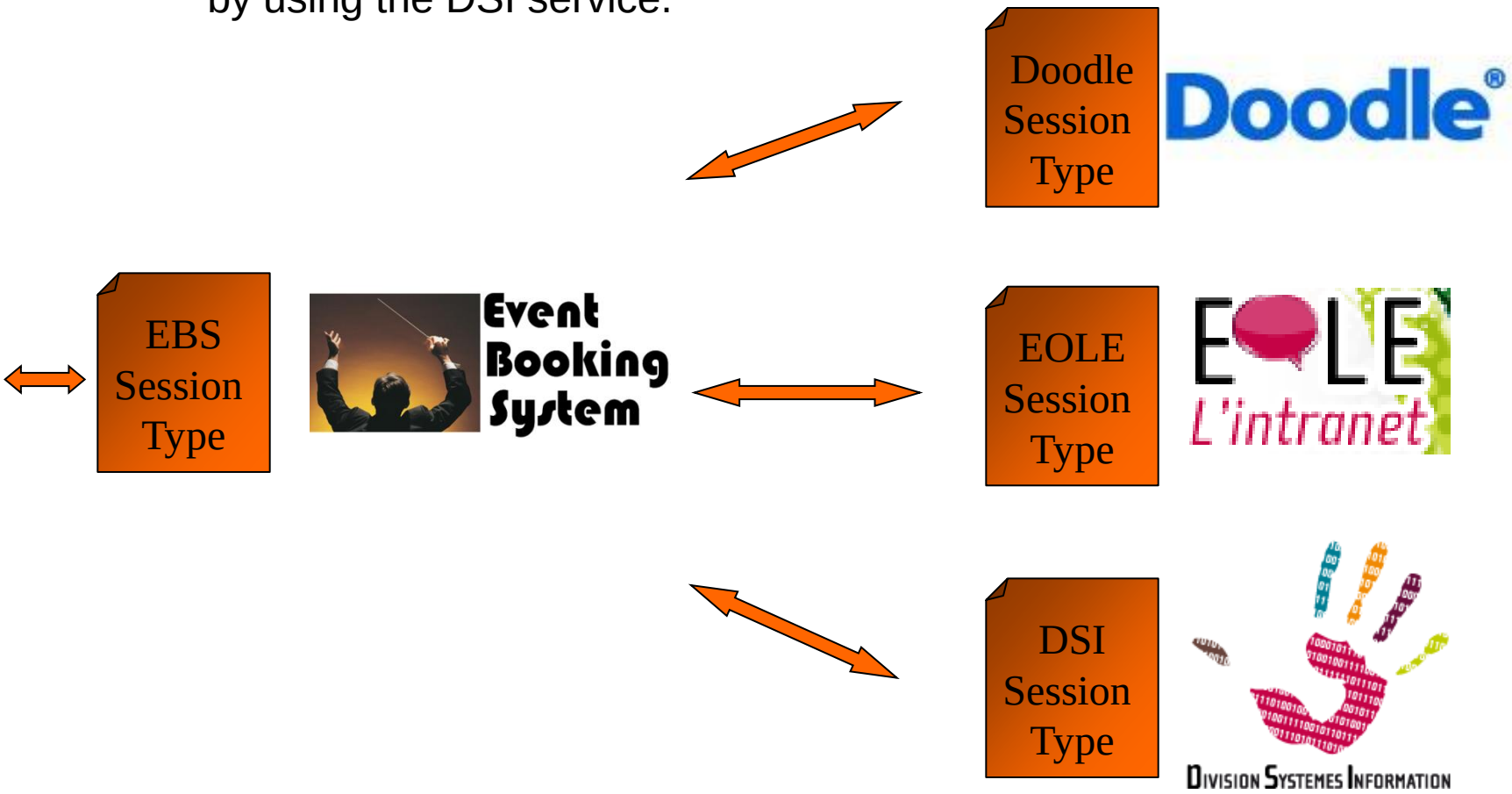
Event Booking System

EOLE L'intranet

Division Systemes Information

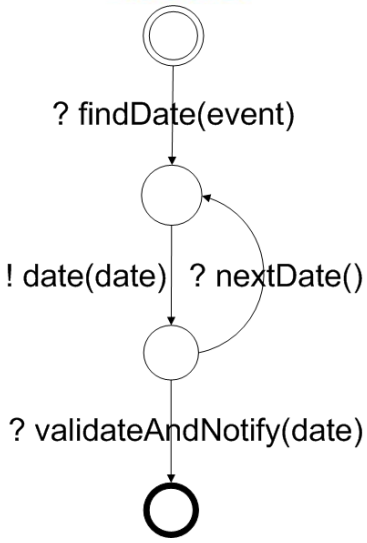
Example service orchestration: The Event Booking System

The Event Booking System chooses a date from a previously set up Doodle page and books a room by using EOLE and some equipment by using the DSI service.

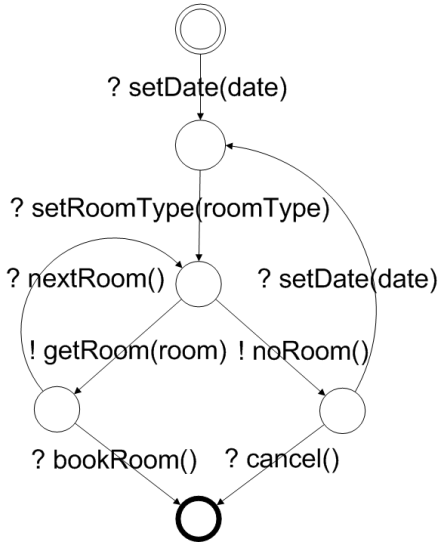


Event Booking System

Doodle

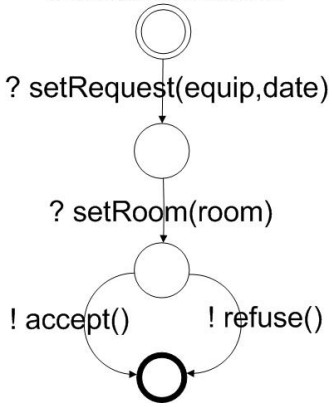


EOLE L'intranet



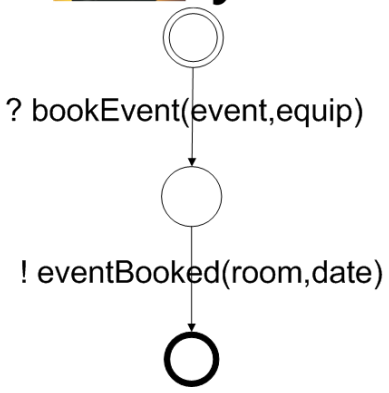
Required Session types

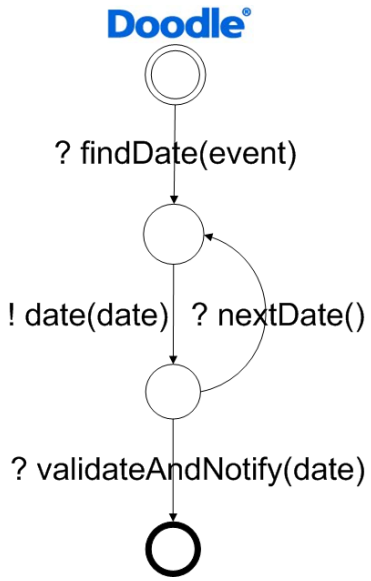
Division Systemes Information

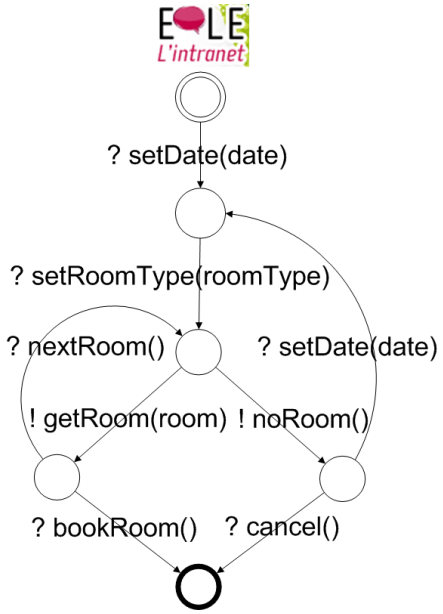


Event Booking System

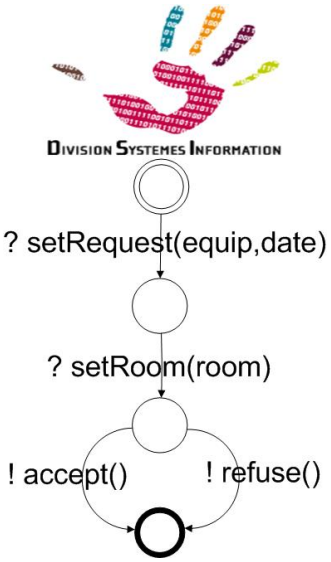
Provided Session type







Required Session types

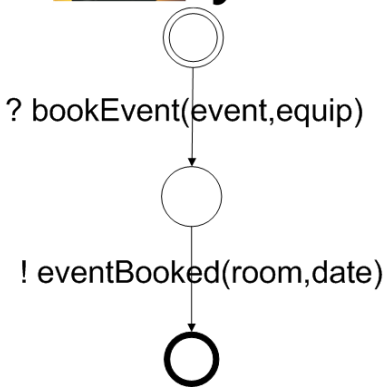


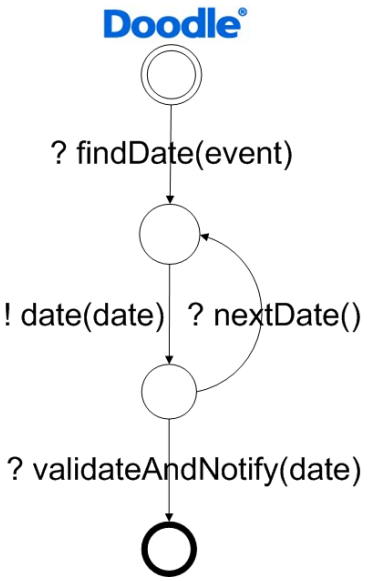
```
Doodle_t doodle = @1;   String date;
EOLE eole = @2;         String room;
DSI dsi = @3;           String event;
EBS_T s;                String equip;
```

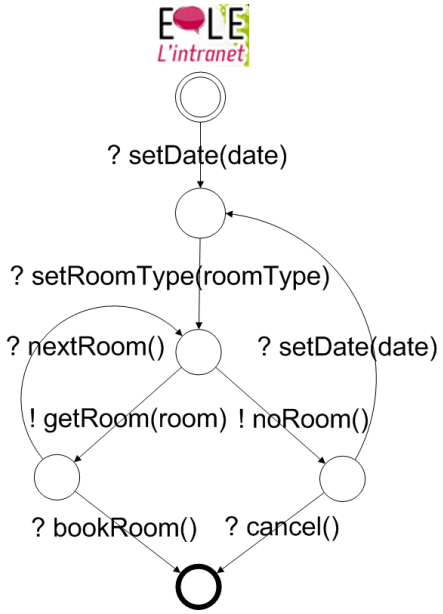
```
s ? bookEvent(event,equip)
```

Event Booking System

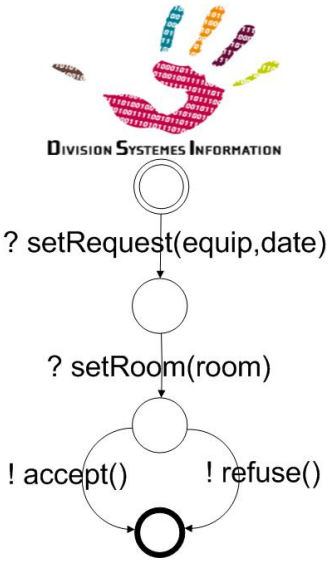
Provided Session type







Required Session types



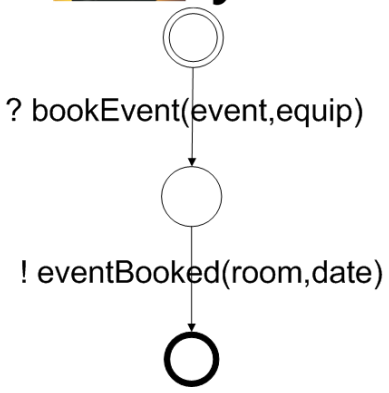
Doodle_t doodle = @1; String date;
EOLE eole = @2; String room;
DSI dsi = @3; String event;
EBS_T s; String equip;

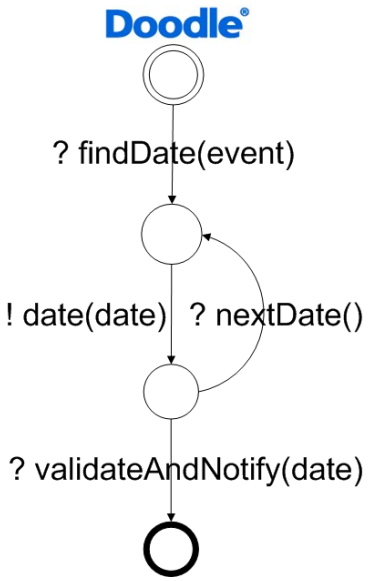
s ? bookEvent(event,equip)

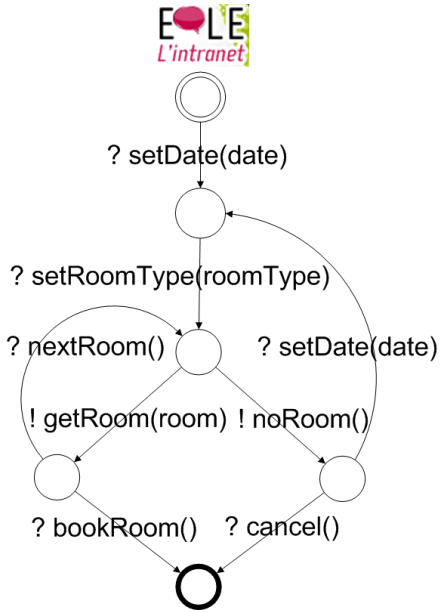
d @ doodle
e @ eole
i @ dsi

PARALLEL

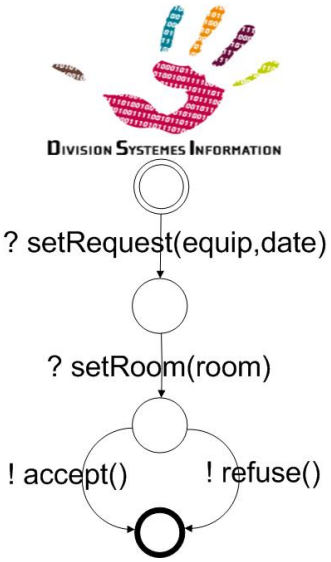
Provided Session type







Required Session types



Doodle_t doodle = @1;
EOLE eole = @2;
DSI dsi = @3;
EBS_T s;

String date;
String room;
String event;
String equip;

s ? bookEvent(event,equip)

d @ doodle
e @ eole
i @ dsi

Event Booking System

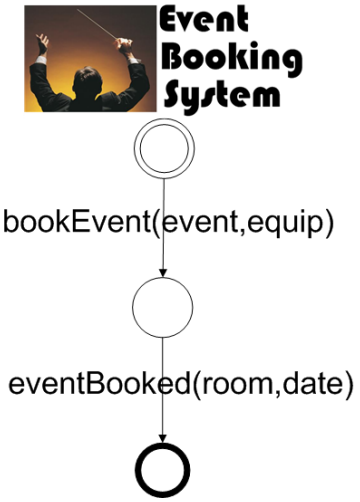
PARALLEL

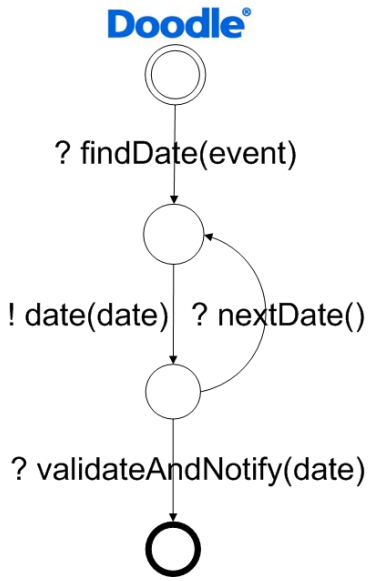
d ! findDate(event)

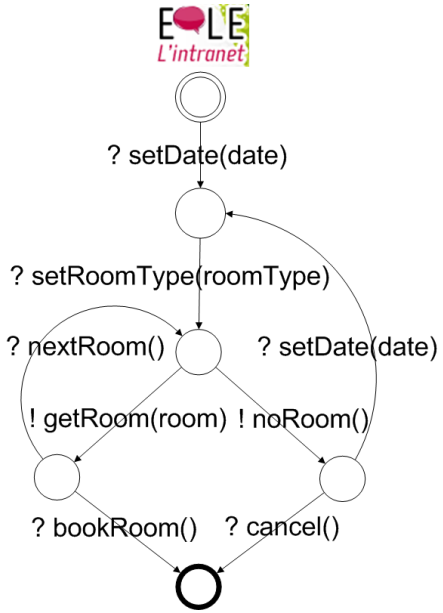
d ? date(date)

d ! validateAndNotify(date)

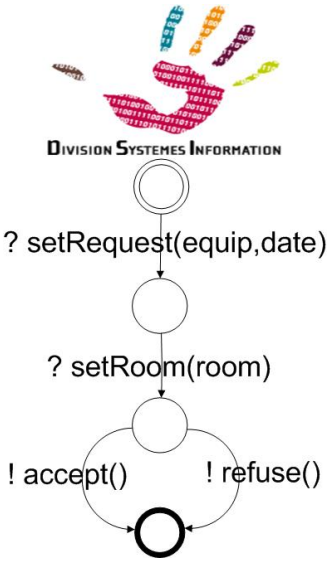
Provided Session type







Required Session types



Doodle_t doodle = @1;
EOLE eole = @2;
DSI dsi = @3;
EBS_T s;

String date;
String room;
String event;
String equip;

s ? bookEvent(event,equip)

d @ doodle
e @ eole
i @ dsi

PARALLEL

d ! findDate(event)

d ? date(date)

d ! validateAndNotify(date)

e ! setDate(date)

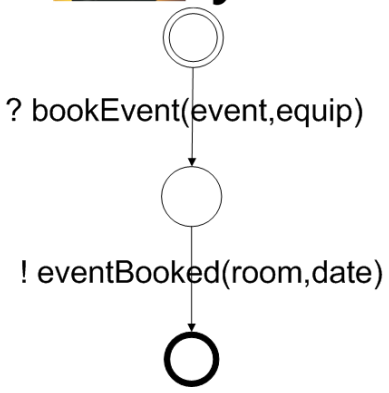
e ! setRoomType(« meeting »)

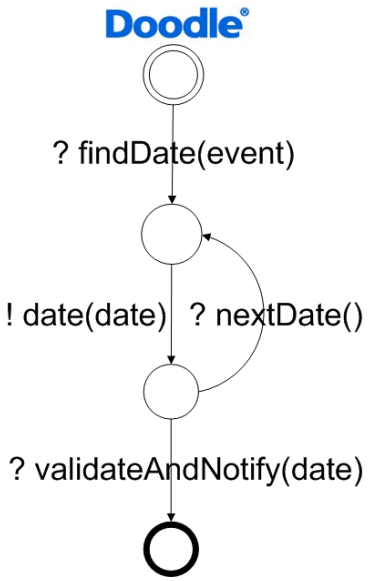
e ? getRoom(room)

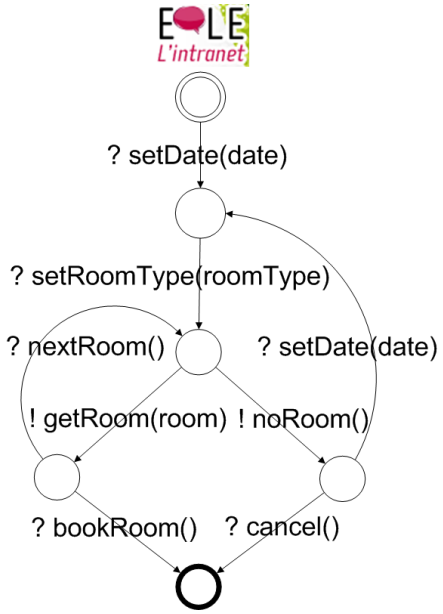
e ? noRoom()

e ! bookRoom()

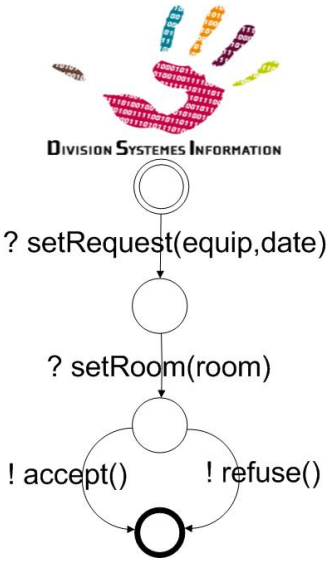
Provided Session type







Required Session types



Doodle_t doodle = @1;
EOLE eole = @2;
DSI dsi = @3;
EBS_T s;

String date;
String room;
String event;
String equip;

s ? bookEvent(event,equip)

d @ doodle
e @ eole
i @ dsi

Event Booking System

PARALLEL

d ! findDate(event)

d ? date(date)

d ! validateAndNotify(date)

e ! setDate(date)

e ! setRoomType(« meeting »)

e ? getRoom(room)

e ? noRoom()

e ! bookRoom()

i ! setRequest(equip,date)

i ! setRoom(room)

i ? accept(room)

i ? refuse()

Provided Session type

? bookEvent(event,equip)

! eventBooked(room,date)

Doodle

? findDate(event)

! date(date)

? nextDate()

? validateAndNotify(date)

EOLE L'intranet

? setDate(date)

? setRoomType(roomType)

? nextRoom()

? setDate(date)

! getRoom(room)

! noRoom()

? bookRoom()

? cancel()

Required Session types

Division Systemes Information

? setRequest(equip,date)

? setRoom(room)

! accept()

! refuse()

Doodle_t doodle = @1;
EOLE eole = @2;
DSI dsi = @3;
EBS_T s;

String date;
String room;
String event;
String equip;

s ? bookEvent(event,equip)

d @ doodle
e @ eole
i @ dsi

Event Booking System

PARALLEL

d ! findDate(event)

d ? date(date)

d ! validateAndNotify(date)

e ! setDate(date)

e ! setRoomType(« meeting »)

e ? getRoom(room)

e ? noRoom()

e ! bookRoom()

i ! setRequest(equip,date)

i ! setRoom(room)

i ? accept(room)

i ? refuse()

s ! eventBooked(date,room)

s ! eventFailed()

Provided Session type

? bookEvent(event,equip)

! eventBooked(room,date)

? findDate(event)

! date(date)

? nextDate()

? validateAndNotify(date)

? setDate(date)

? setRoomType(roomType)

? nextRoom()

? setDate(date)

! getRoom(room)

! noRoom()

? bookRoom()

? cancel()

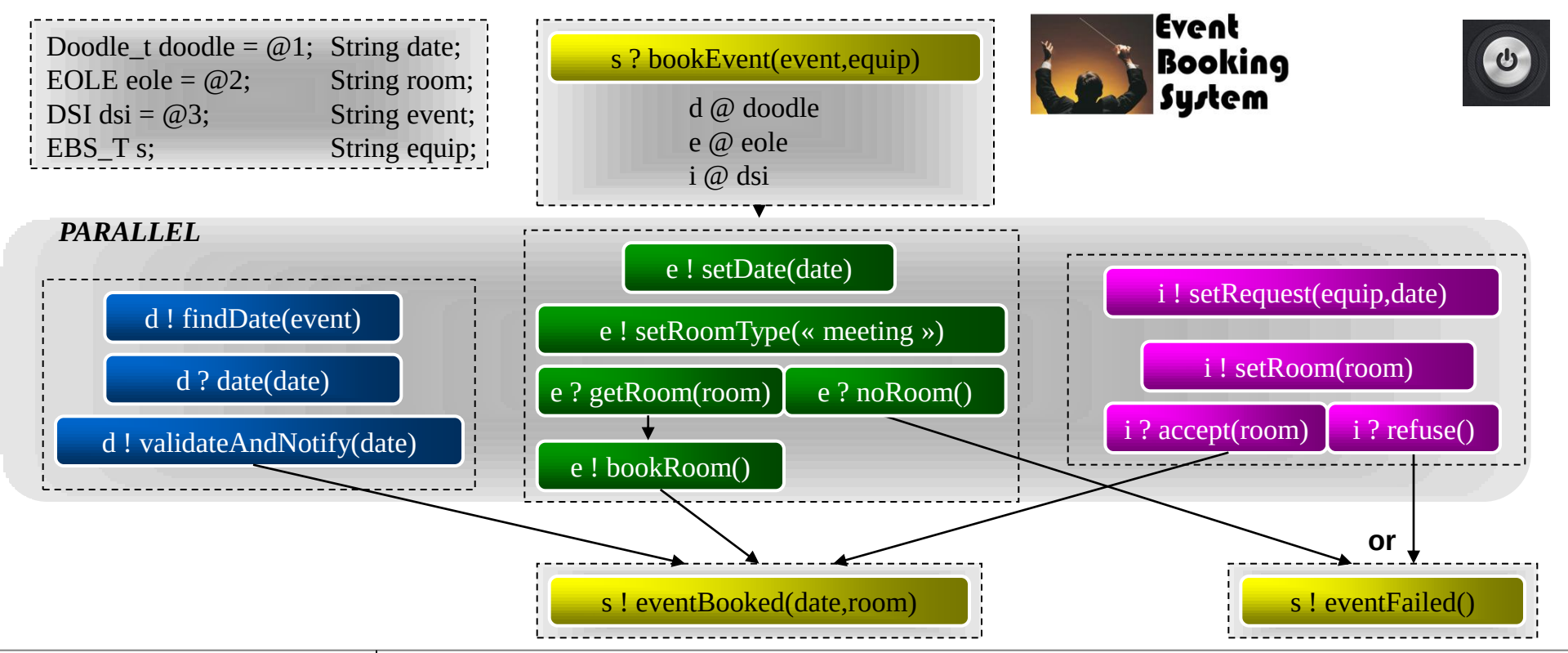
Required Session types

? setRequest(equip,date)

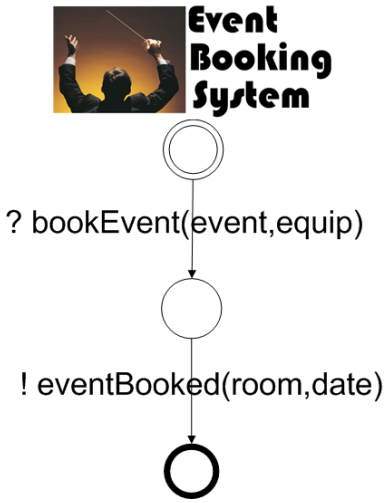
? setRoom(room)

! accept()

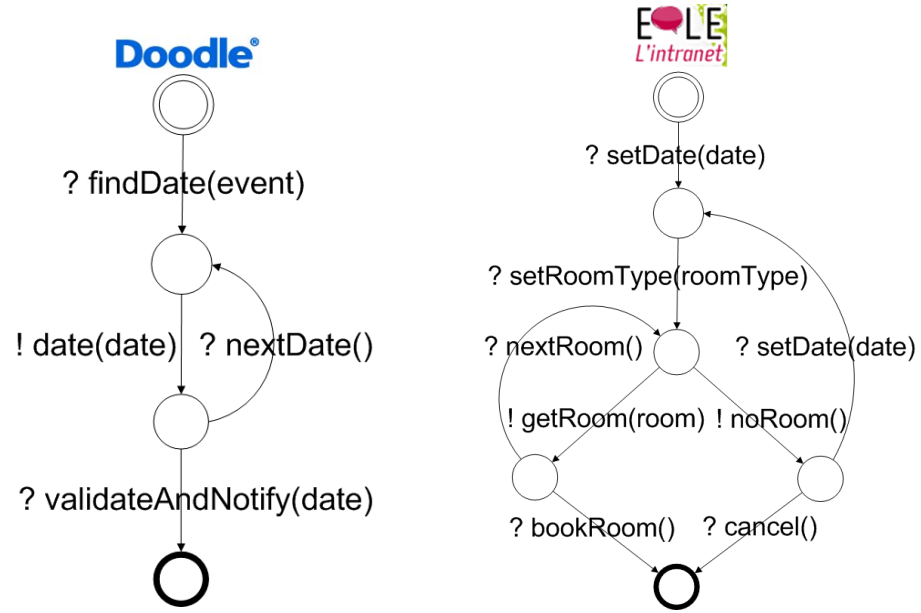
! refuse()



Provided Session type



Required Session types



```
Doodle_t doodle = @1;   String date;
EOLE eole = @2;         String room;
DSI dsi = @3;           String event;
EBS_T s;                String equip;
```

s ? bookEvent(event,equip)

d @ doodle
e @ eole
i @ dsi

Event Booking System

What happens if we push the button ?

PARALLEL

d ! findDate(event)

d ? date(date)

d ! validateAndNotify(date)

e ! setDate(date)

e ! setRoomType(« meeting »)

e ? getRoom(room) e ? noRoom()

e ! bookRoom()

i ! setRequest(equip,date)

i ! setRoom(room)

i ? accept(room) i ? refuse()

or

s ! eventBooked(date,room)

s ! eventFailed()

Provided Session type

Event Booking System

? bookEvent(event,equip)

! eventBooked(room,date)

Required Session types

Doodle

? findDate(event)

! date(date) ? nextDate()

? validateAndNotify(date)

EOLE L'intranet

? setDate(date)

? setRoomType(roomType)

? nextRoom() ? setDate(date)

! getRoom(room) ! noRoom()

? bookRoom() ? cancel()

Division Systemes Information

? setRequest(equip,date)

? setRoom(room)

! accept() ! refuse()

Doodle_t doodle = @1; String date;
EOLE eole = @2; String room;
DSI dsi = @3; String event;
EBS_T s; String equip;

s ? bookEvent(event,equip)

d @ doodle
e @ eole
i @ dsi

4 errors detected

PARALLEL

d ! findDate(event)
d ? date(date)
d ! validateAndNotify(date)

e ! setDate(date)
e ! setRoomType(« meeting »)
e ? getRoom(room) | e ? noRoom()
e ! bookRoom()

i ! setRequest(equip,date)
i ! setRoom(room)
i ? accept(room) | i ? refuse()

or

s ! eventBooked(date,room)

s ! eventFailed()

Provided Session type

? bookEvent(event,equip)

! eventBooked(room,date)

Required Session types

Doodle

? findDate(event)

! date(date) ? nextDate()

? validateAndNotify(date)

EOLE L'intranet

? setDate(date)

? setRoomType(roomType)

? nextRoom() ? setDate(date)

! getRoom(room) ! noRoom()

? bookRoom() ? cancel()

Division Systemes Information

? setRequest(equip,date)

? setRoom(room)

! accept() ! refuse()

Event Booking System

3 variable errors

Doodle_t doodle = @1;
EOLE eole = @2;
DSI dsi = @3;
EBS_T s;

String date;
String room;
String event;
String equip;

s ? bookEvent(event,equip)

d @ doodle
e @ eole
i @ dsi

PARALLEL

d ! findDate(event)

d ? date(date)

d ! validateAndNotify(date)

e ! setDate(date)

e ! setRoomType(« meeting »)

e ? getRoom(room)

e ? noRoom()

e ! bookRoom()

i ! setRequest(equip,date)

i ! setRoom(room)

i ? accept(room)

i ? refuse()

or

s ! eventBooked(date,room)

s ! eventFailed()

Event Booking System

3 variable errors

```
Doodle_t doodle = @1;   String date;
EOLE eole = @2;         String room;
DSI dsi = @3;           String event;
EBS_T s;                String equip;
```

s ? bookEvent(event,equip)

d @ doodle
e @ eole
i @ dsi

PARALLEL

d ! findDate(event)

d ? date(date)

d ! validateAndNotify(date)

e ! setDate(date)

e ! setRoomType(« meeting »)

e ? getRoom(room)

e ? noRoom()

e ! bookRoom()

i ! setRequest(equip,date)

i ! setRoom(room)

i ? accept(room)

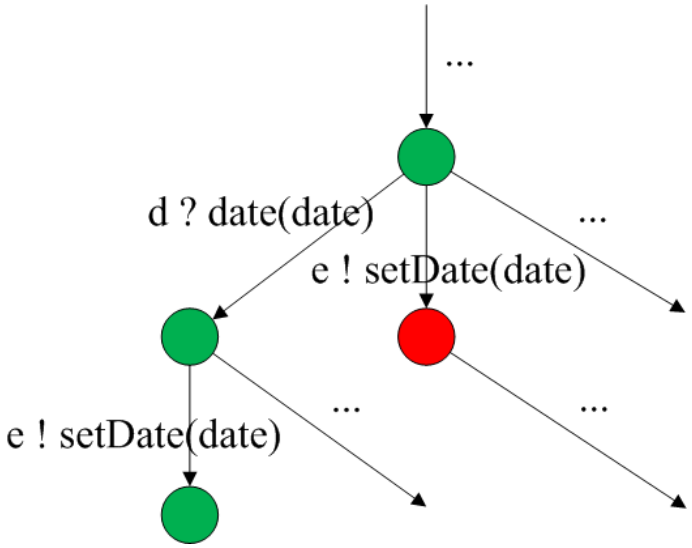
i ? refuse()

or

s ! eventBooked(date,room)

s ! eventFailed()

Part of the Event Booking System's ***control graph***.



= valid state

= error state

Event Booking System

1 error left

Doodle_t doodle = @1; String date;
EOLE eole = @2; String room;
DSI dsi = @3; String event;
EBS_T s; String equip;

s ? bookEvent(event,equip)

d @ doodle
e @ eole
i @ dsi

PARALLEL

d ! findDate(event)

d ? date(date)

d ! validateAndNotify(date)

e ! setDate(date)

e ! setRoomType(« meeting »)

e ? getRoom(room) | e ? noRoom()

e ! bookRoom()

i ! setRequest(equip,date)

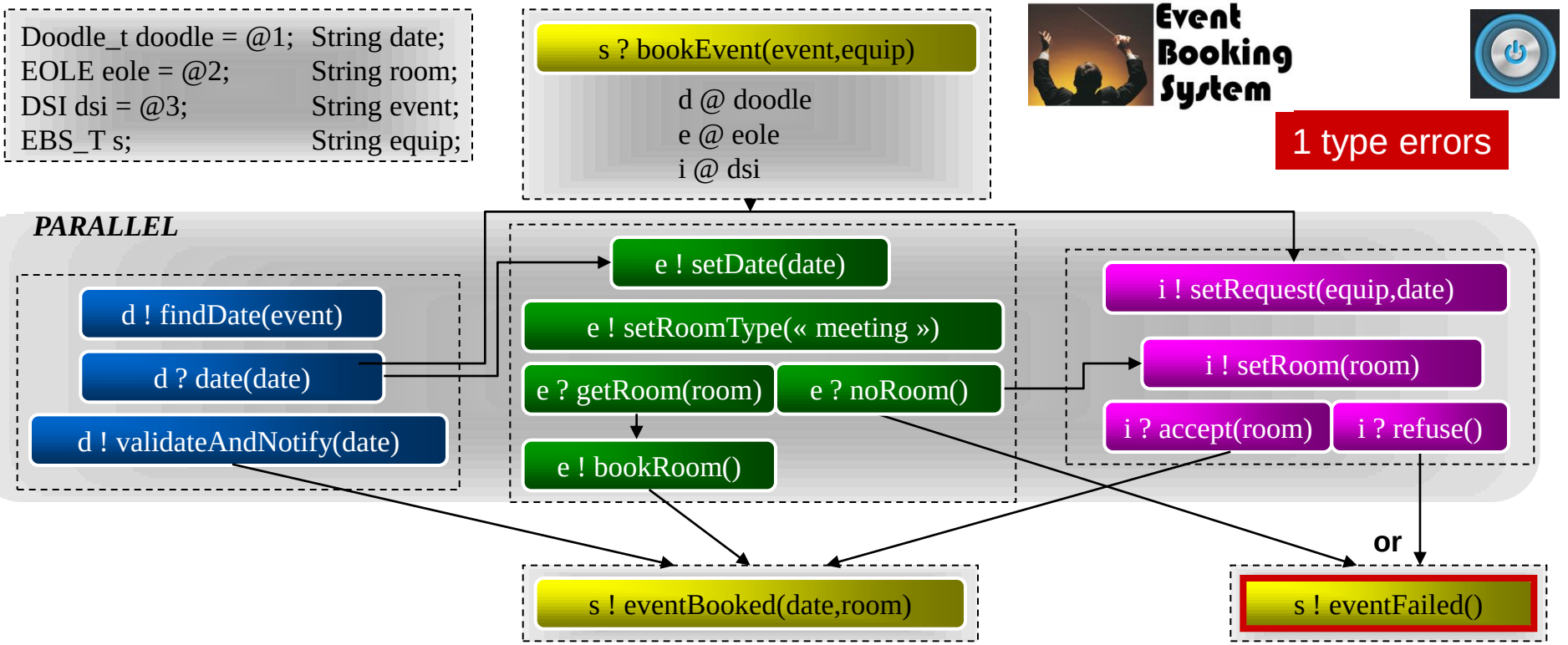
i ! setRoom(room)

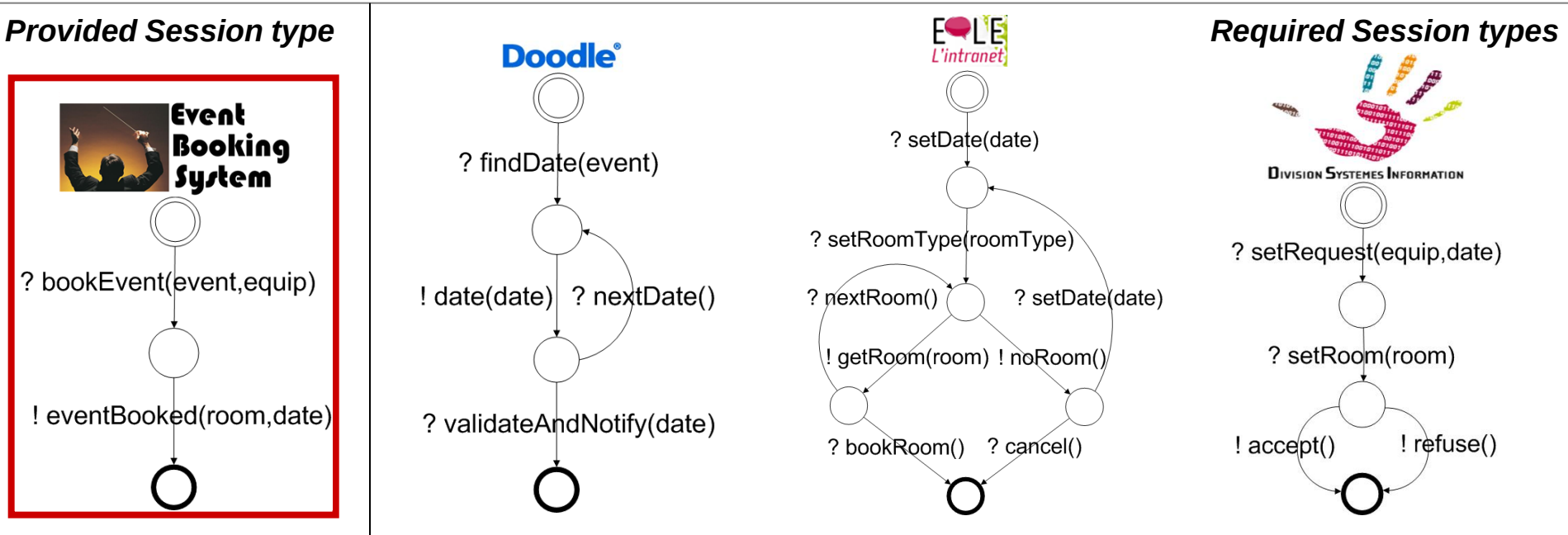
i ? accept(room) | i ? refuse()

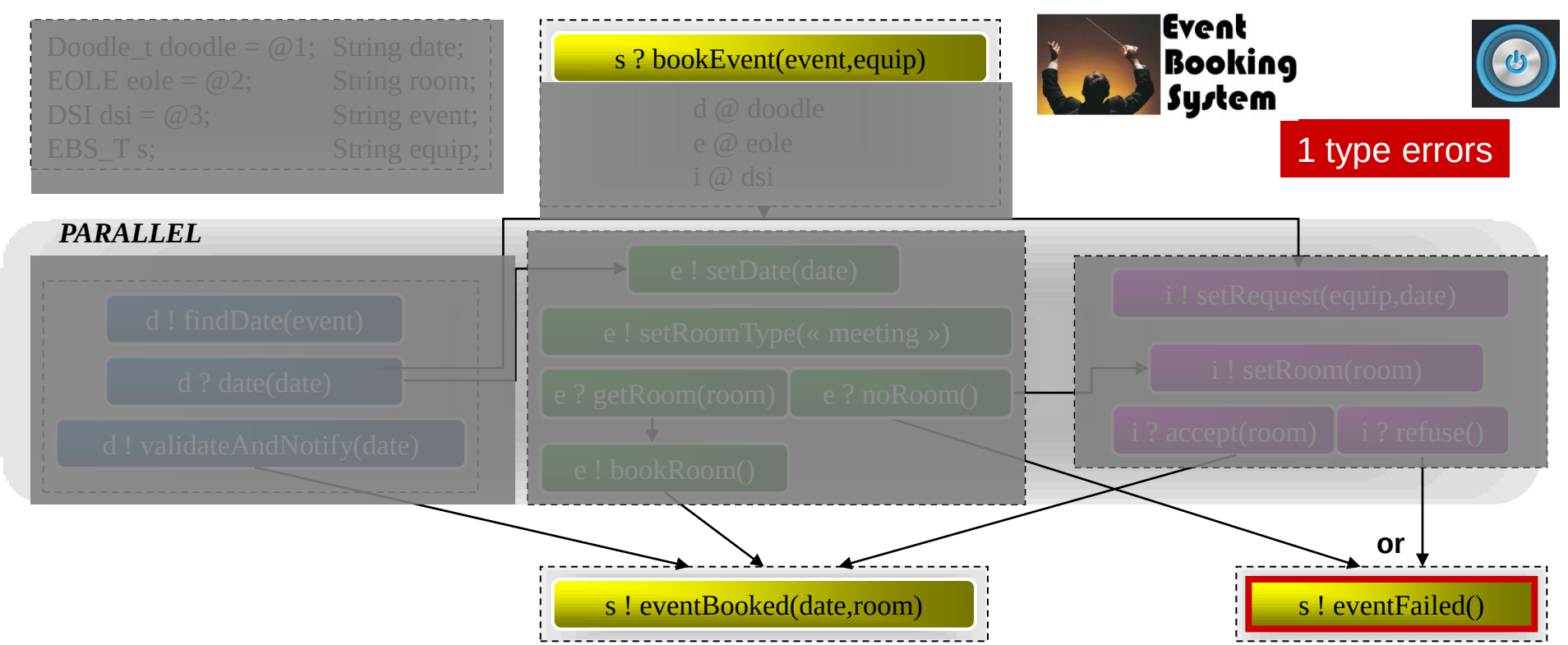
or

s ! eventBooked(date,room)

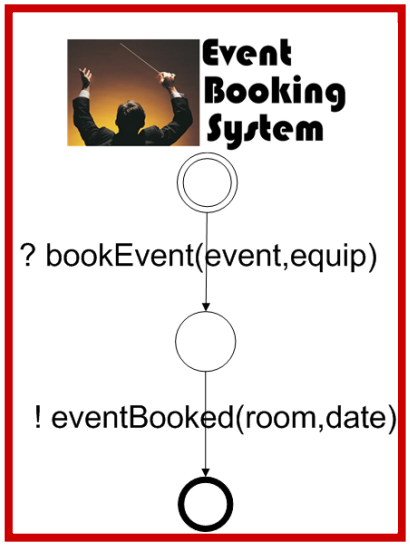
s ! eventFailed()


Doodle_t doodle = @1;
EOLE eole = @2;
DSI dsi = @3;
EBS_T s;
String date;
String room;
String event;
String equip;
s ? bookEvent(event,equip)
d @ doodle
e @ eole
i @ dsi
Event Booking System
1 type errors
PARALLEL
d ! findDate(event)
d ? date(date)
d ! validateAndNotify(date)
e ! setDate(date)
e ! setRoomType(« meeting »)
e ? getRoom(room)
e ? noRoom()
e ! bookRoom()
i ! setRequest(equip,date)
i ! setRoom(room)
i ? accept(room)
i ? refuse()
or
s ! eventBooked(date,room)
s ! eventFailed()


Provided Session type
Event Booking System
? bookEvent(event,equip)
! eventBooked(room,date)
Doodle
? findDate(event)
! date(date)
? nextDate()
? validateAndNotify(date)
EOLE L'intranet
? setDate(date)
? setRoomType(roomType)
? nextRoom()
? setDate(date)
! getRoom(room)
! noRoom()
? bookRoom()
? cancel()
Required Session types
Division Systemes Information
? setRequest(equip,date)
? setRoom(room)
! accept()
! refuse()

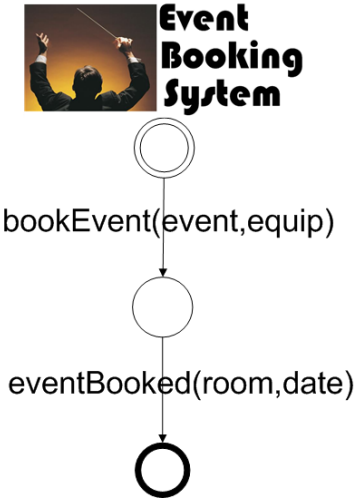


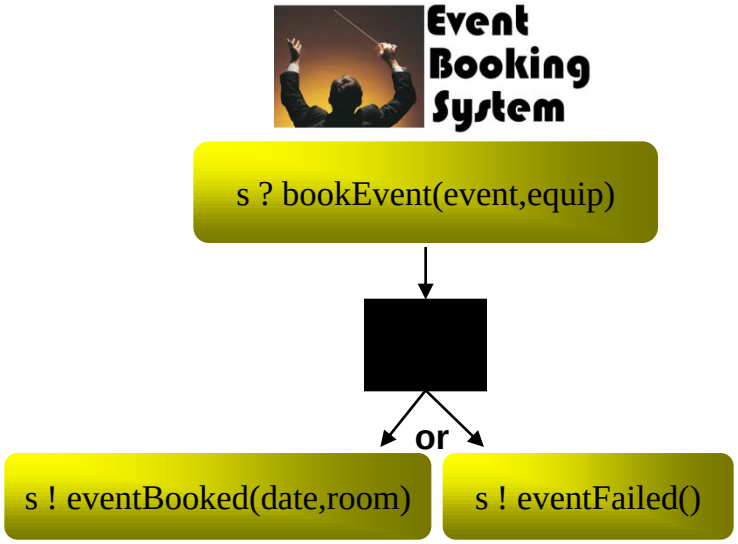
Provided Session type

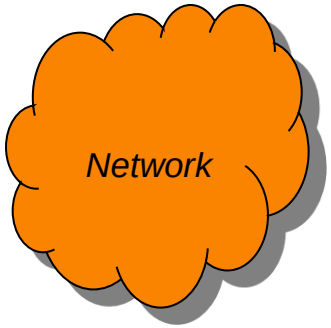


Web service client

b @ EBS
b ! bookEvent(« dbweb seminar », « laptop »)
b ? eventBooked(room,date)

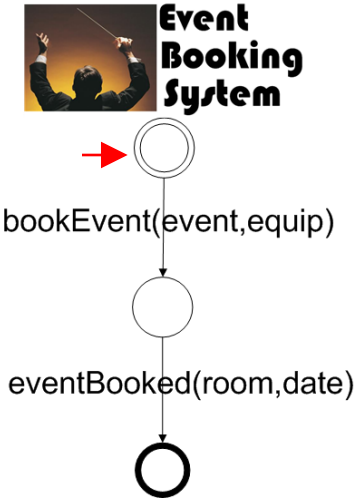


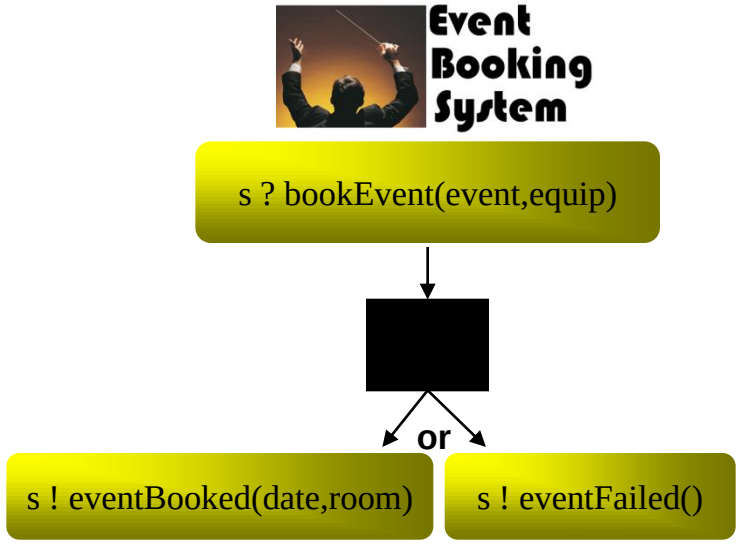




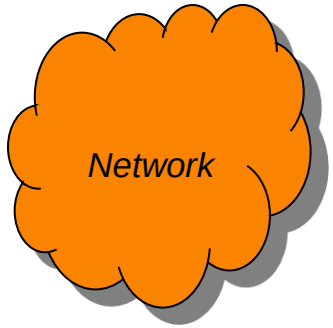
Web service client

b @ EBS
b ! bookEvent(« dbweb seminar », « laptop »)
b ? eventBooked(room,date)



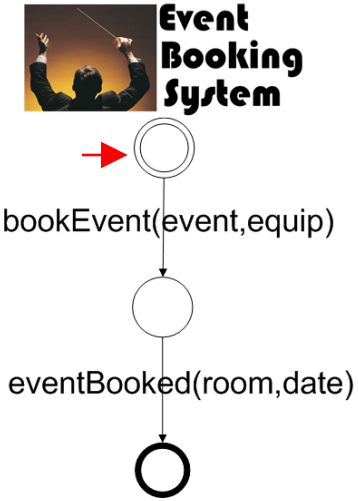


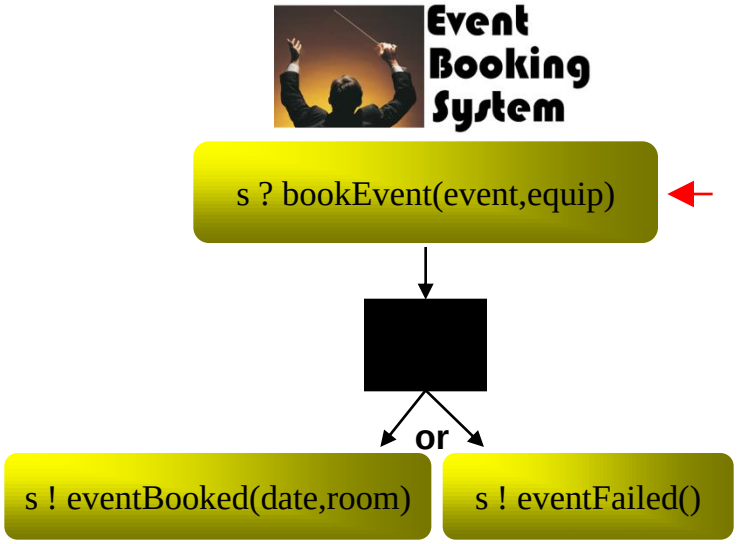
b @ EBS



Web service client

b @ EBS
b ! bookEvent(« dbweb seminar », « laptop »)
b ? eventBooked(room,date)





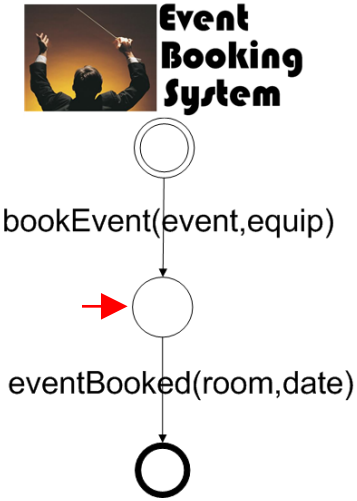
b @ EBS

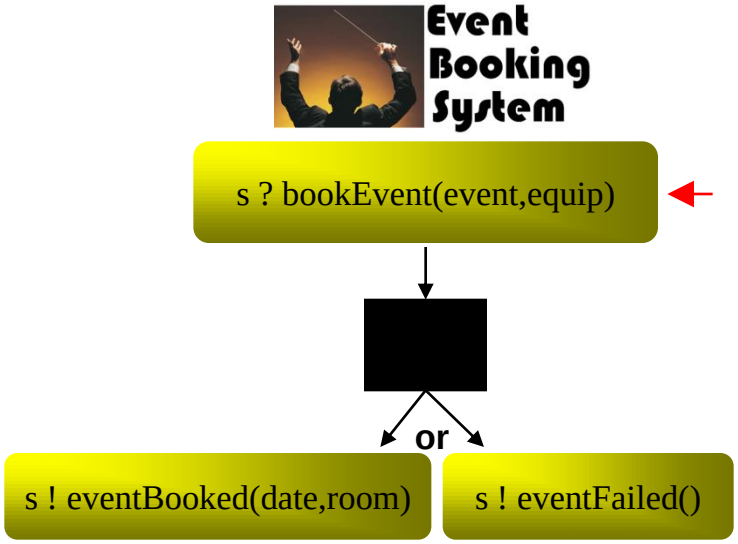
b @ EBS

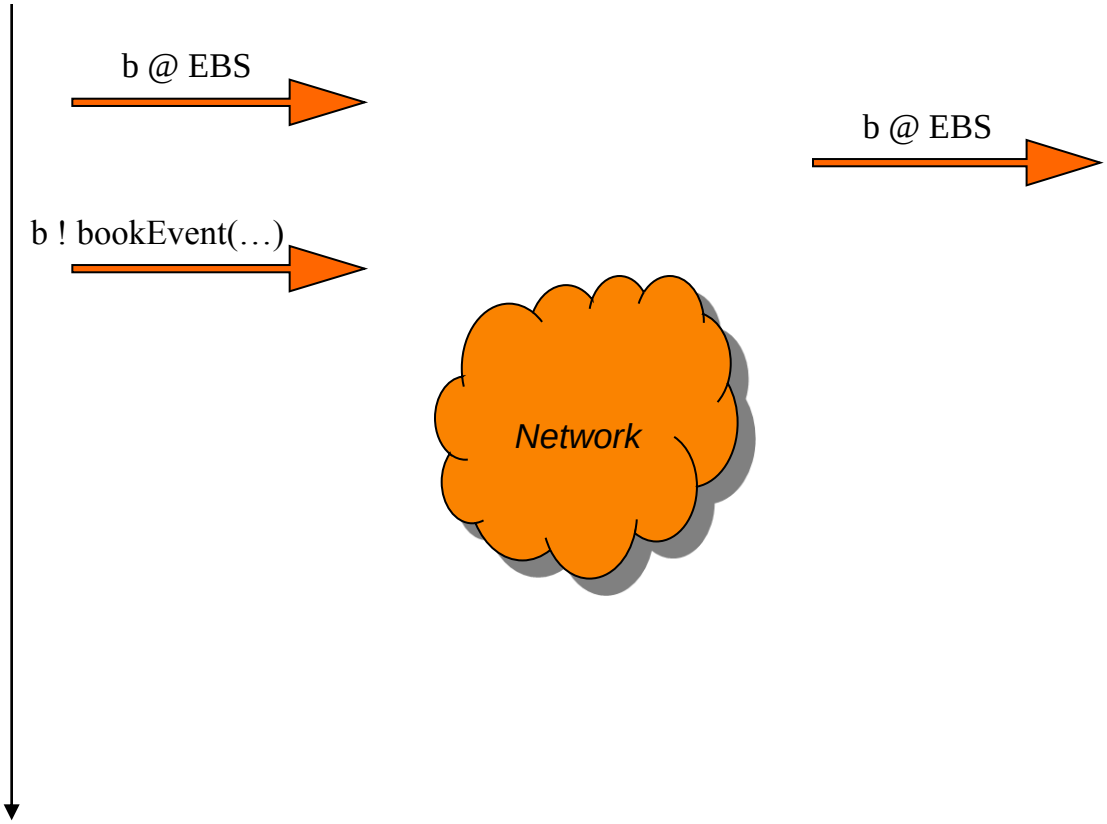
Network

Web service client

b @ EBS
b ! bookEvent(« dbweb seminar », « laptop »)
b ? eventBooked(room,date)

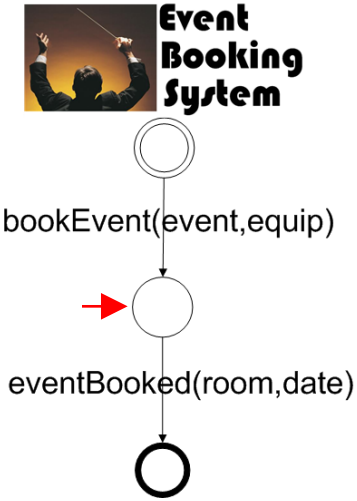


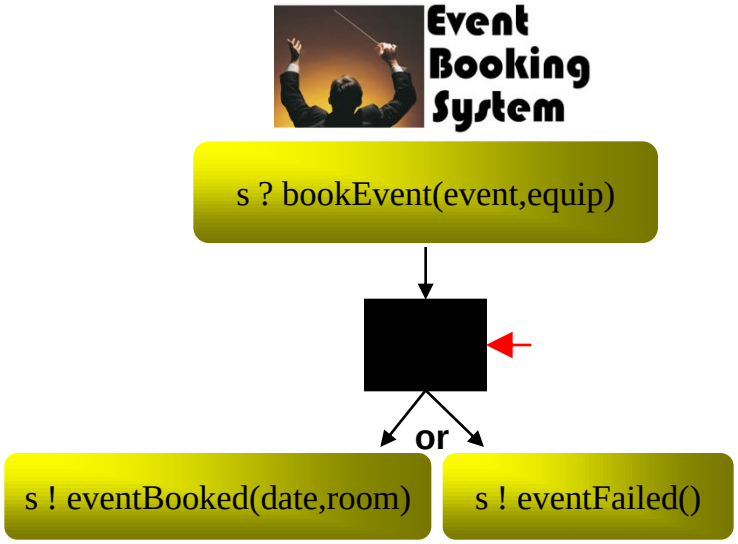


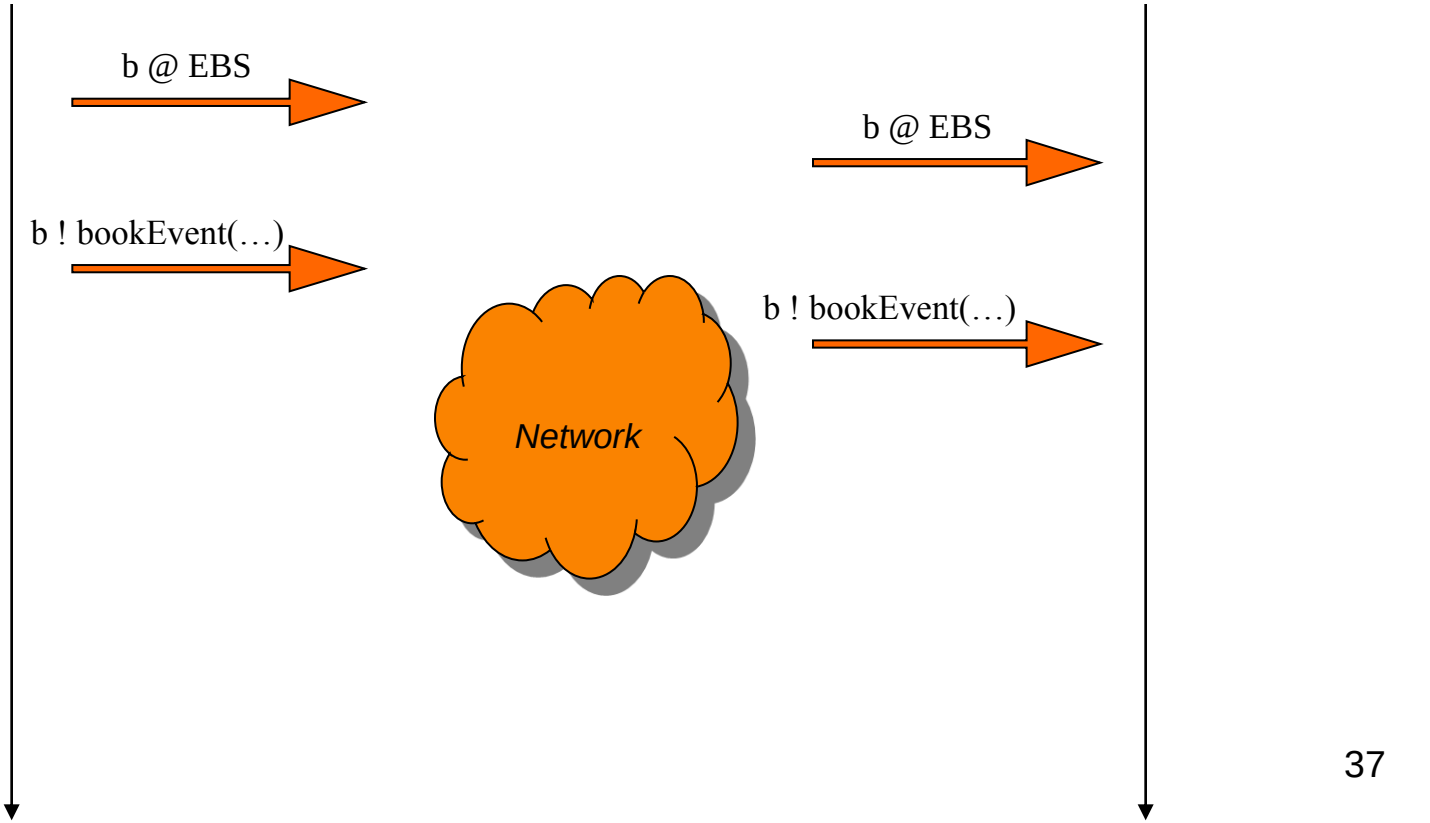


Web service client

b @ EBS
b ! bookEvent(« dbweb seminar », « laptop »)
b ? eventBooked(room,date)

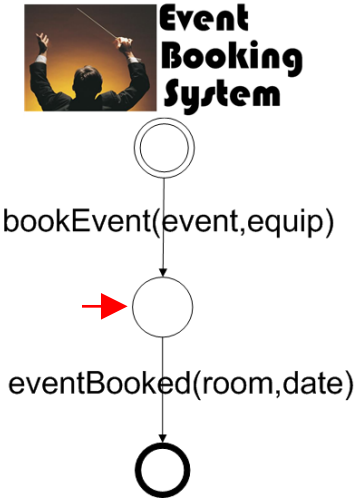


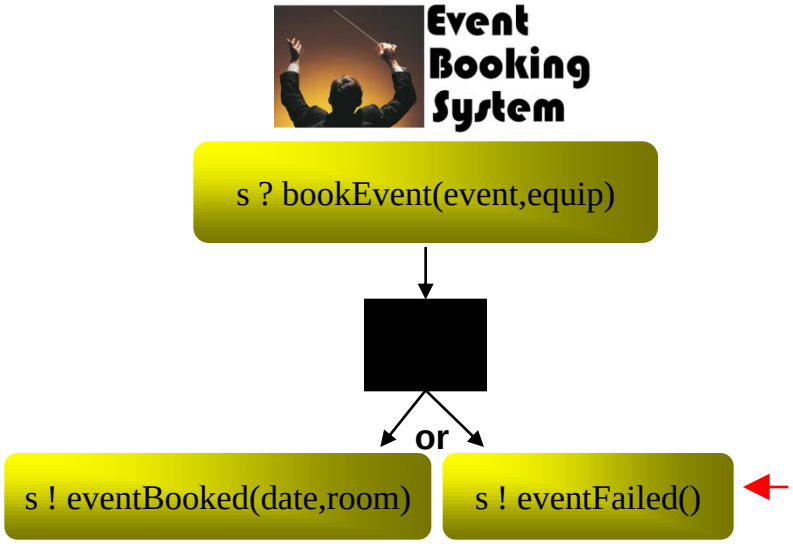


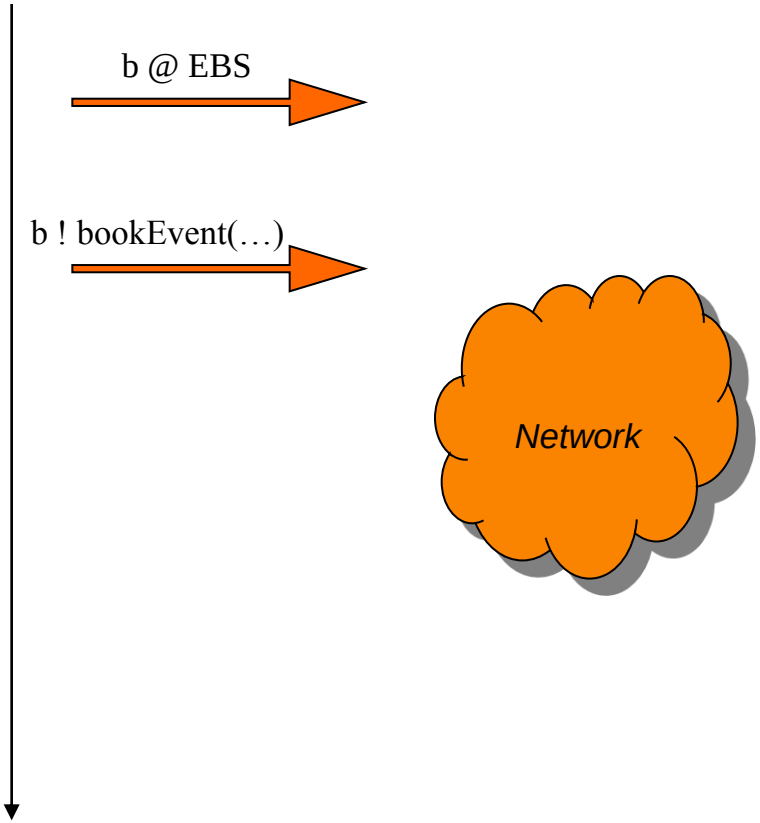


Web service client

b @ EBS
b ! bookEvent(« dbweb seminar », « laptop »)
b ? eventBooked(room,date)

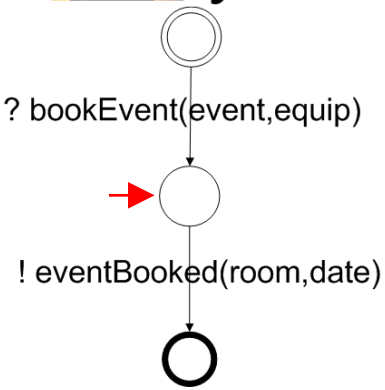


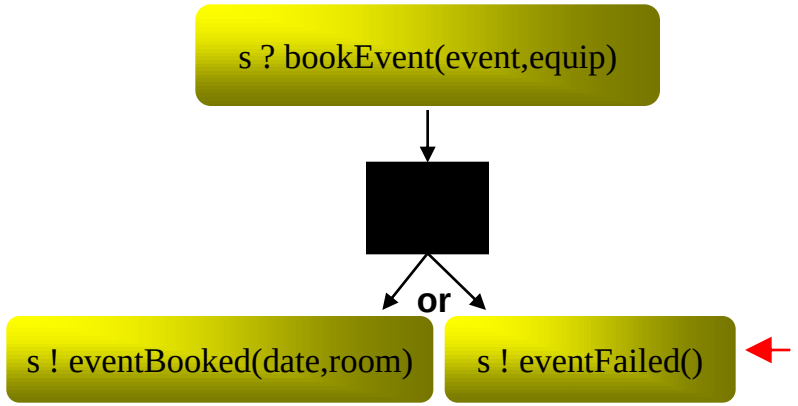


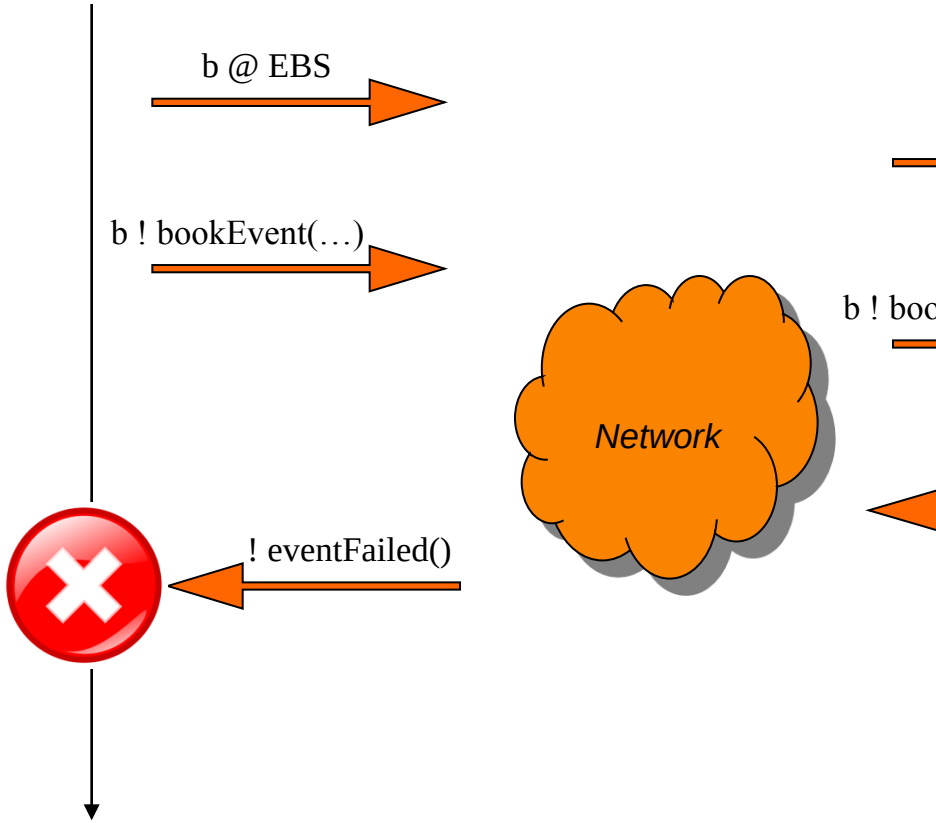


Web service client

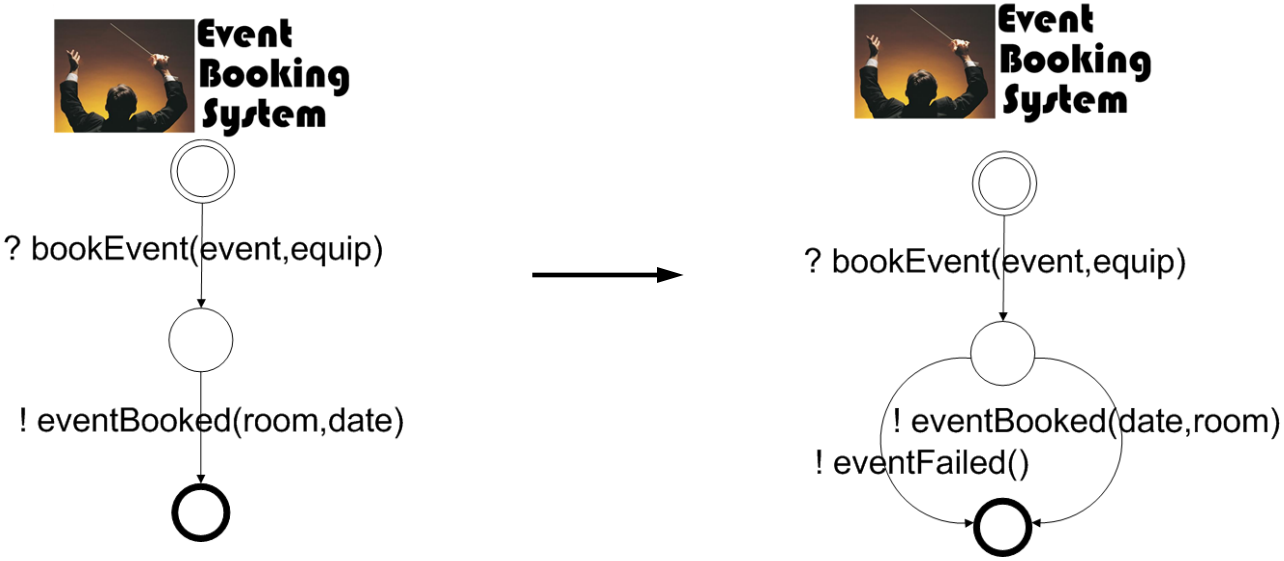
b @ EBS
b ! bookEvent(« dbweb seminar », « laptop »)
b ? eventBooked(room,date)

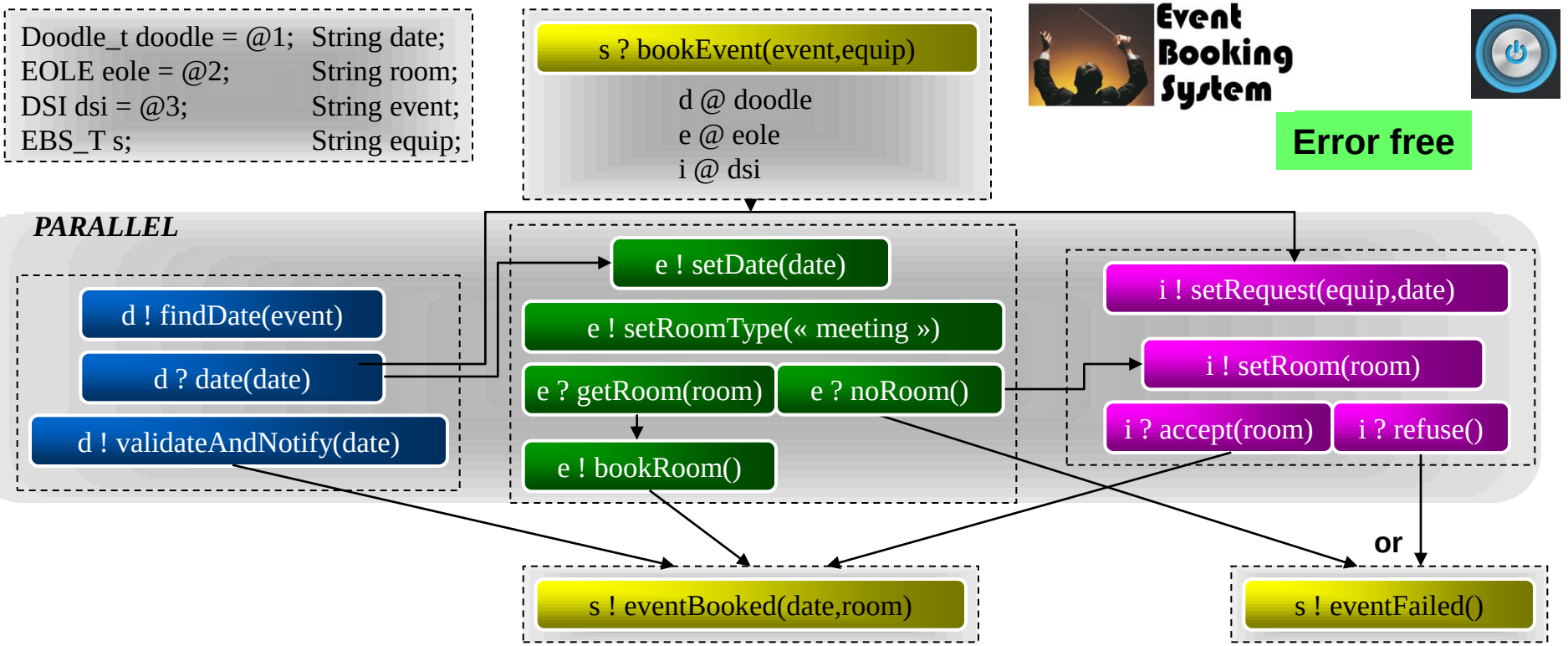




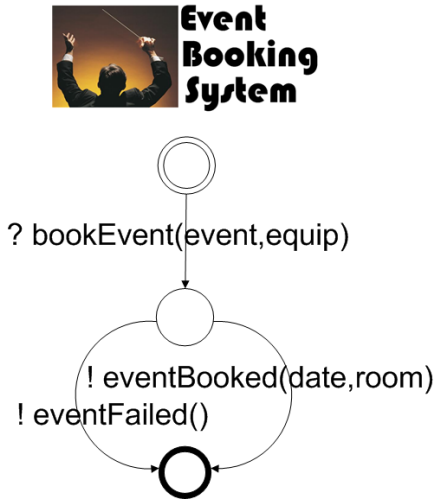


Solution:

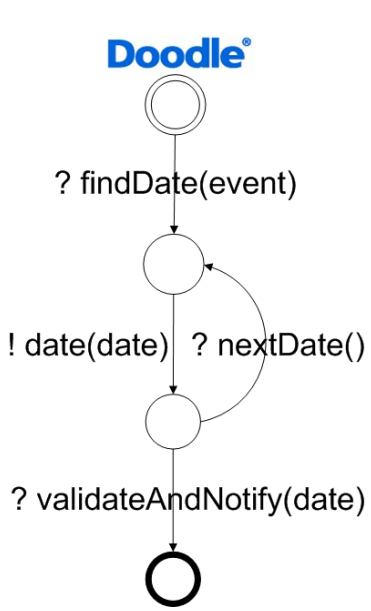

Event
Booking
System
? bookEvent(event,equip)
! eventBooked(room,date)
Event
Booking
System
? bookEvent(event,equip)
! eventBooked(date,room)
! eventFailed()

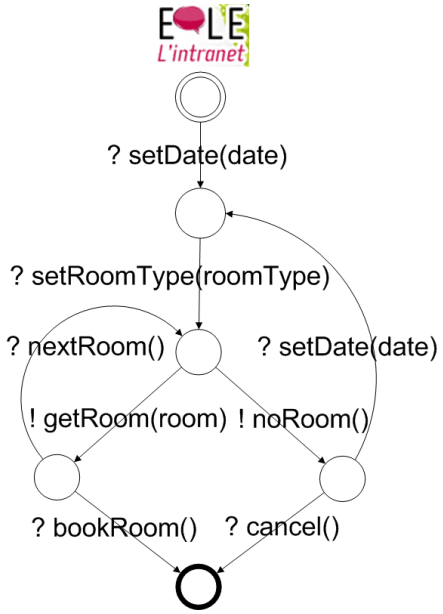


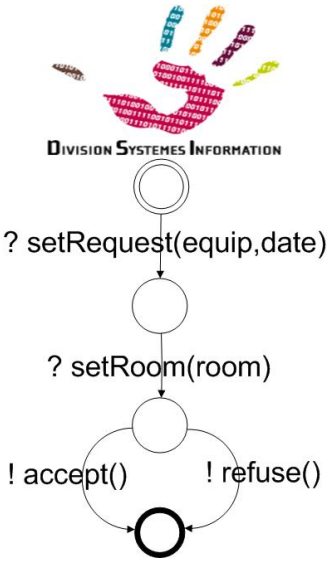
Provided Session type



Required Session types







Related work

- The language I presented is based on BPEL
  - Much work has gone into giving BPEL formal semantics, not an easy task
- The session paradigm is a vast topic that has warranted much attention recently
  - See “Language primitives and type discipline for structured communication-based programming” – Kohei Honda, Vasco T. Vasconcelos, Makoto Kubo.
  - Interesting extensions such as multiparty sessions

Main contributions

- A formal definition of an orchestration language
  - Based on BPEL
  - Novel semantics that capture important parts of BPEL compared to related work
- Behavioural typing extension
  - Session types are supported
- Main result: ***Theorem of interaction safety*** – *any well-typed configuration of services is interaction safe.*

Conclusion

- Web service orchestrations are value added compositions of existing services
- Orchestrations are prone to certain types of runtime errors
- Session-based programming is a new paradigm that can help to detect these errors before deployment
- **We can prove that if services correctly implement their session types, then the property of safe interaction is verified.**

Thank you for listening

Remember : ***Just sessionize it !***