UML'02 – 04/10/2002

The Specification of UML Collaborations as Interaction Components

Éric CARIOU Antoine BEUGNARD

ENST Bretagne

GOALS

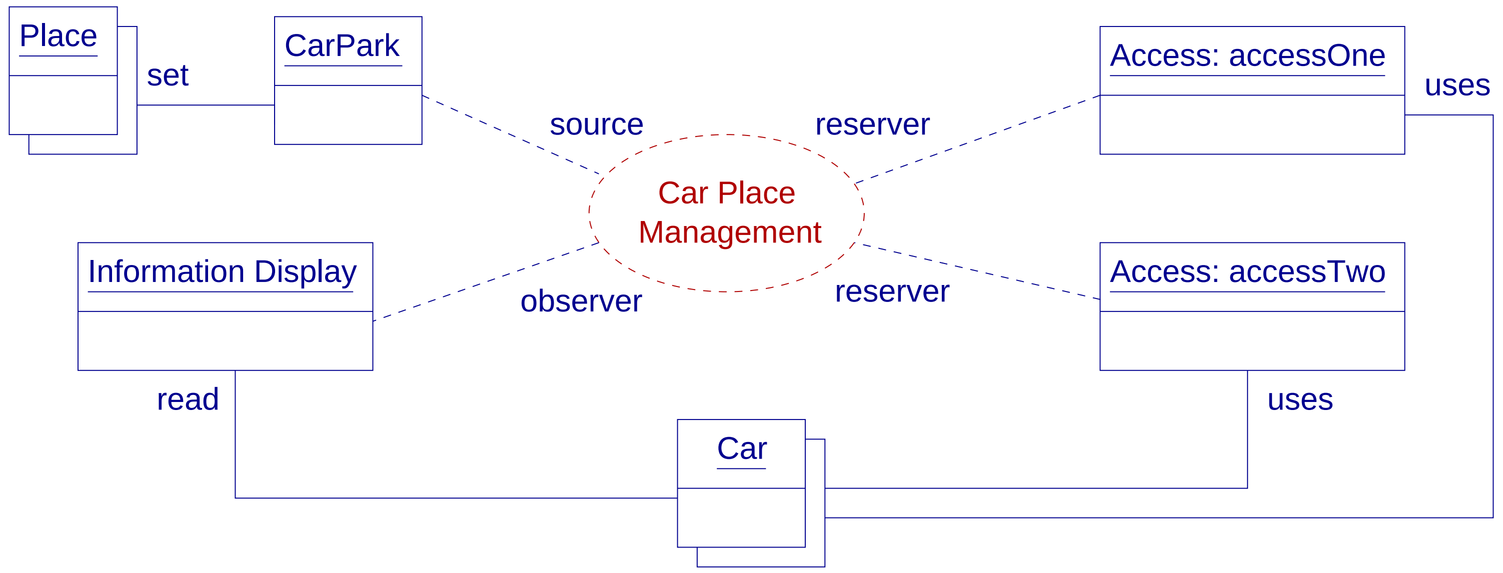
- ➤ Collaboration diagrams: core feature of UML
- ➤ Define an interaction abstraction at specification level
- ➤ But how to have interaction abstractions at implementation level?
  - ➥ Classicaly, no more trace of these abstractions at this level
  - ➥ Communication only through “low-level” primitives (RPC)
- ➤ Our proposition:
  - ➥ The reification of interaction abstractions as software components
  - ➥ Methodology of UML specification of these components

OUTLINE

1. Study of a collaboration in a reservation application
2. A more interesting way to specify the collaboration
3. Introduction to interaction components
4. Methodology of interaction component specification in UML

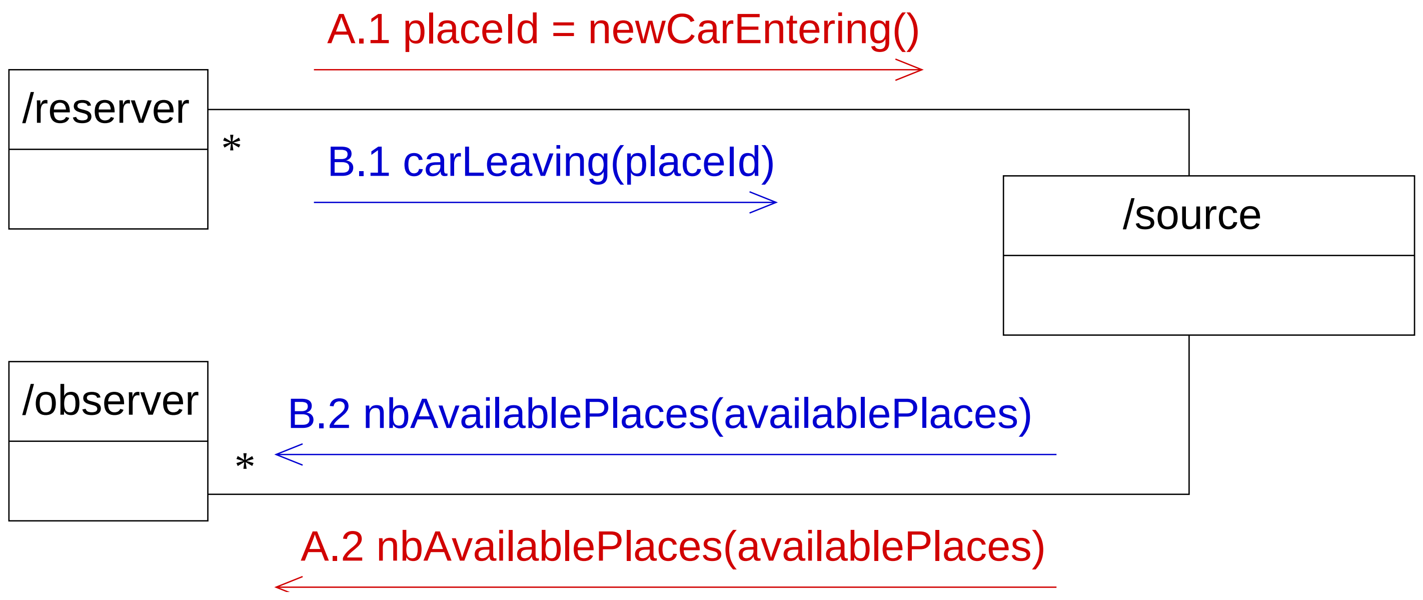
A Car Park Management System

- A car park with two accesses
- A display shows the number of available places



- The four components interact through a UML collaboration

THE COLLABORATION USED


A.1 placeId = newCarEntering()
/reserver
*
B.1 carLeaving(placeId)
/source
/observer
B.2 nbAvailablePlaces(availablePlaces)
*
A.2 nbAvailablePlaces(availablePlaces)

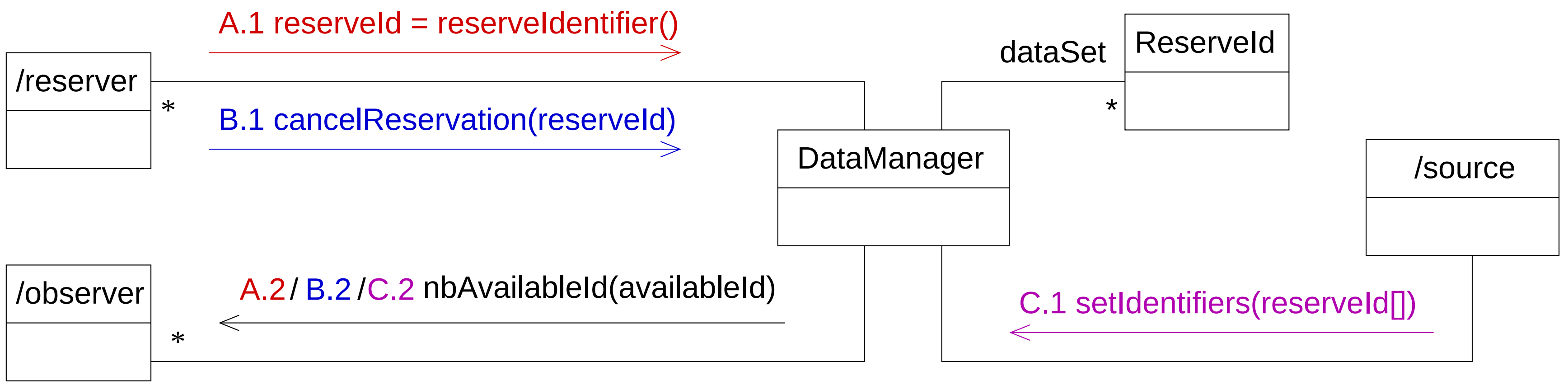
- Simply shows messages sent among roles

REUSE OF THE COLLABORATION

- ➤ With minor changes:
  - ➥ If the source is an airline company
  - ➥ If reservers and observers are travel agencies
  - ⇒ reservation of places in a flight
- ➤ Warning: the data management system must be implemented by the source role

⇒ Why not putting this system into the collaboration?

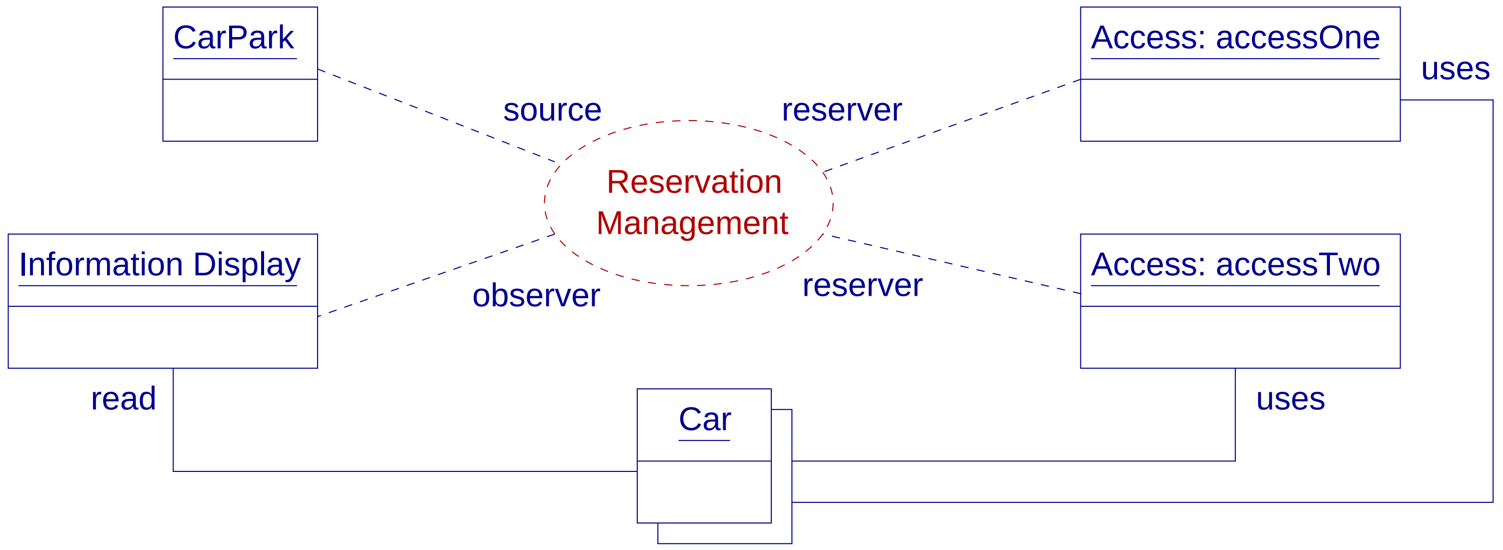
NEW COLLABORATION DESIGN



A.1 reserveId = reserveIdentifier()
/reserver
*
B.1 cancelReservation(reserveId)
dataSet
ReserveId
*
DataManager
/source
/observer
A.2 / B.2 /C.2 nbAvailableId(availableId)
*
C.1 setIdentifiers(reserveId[])

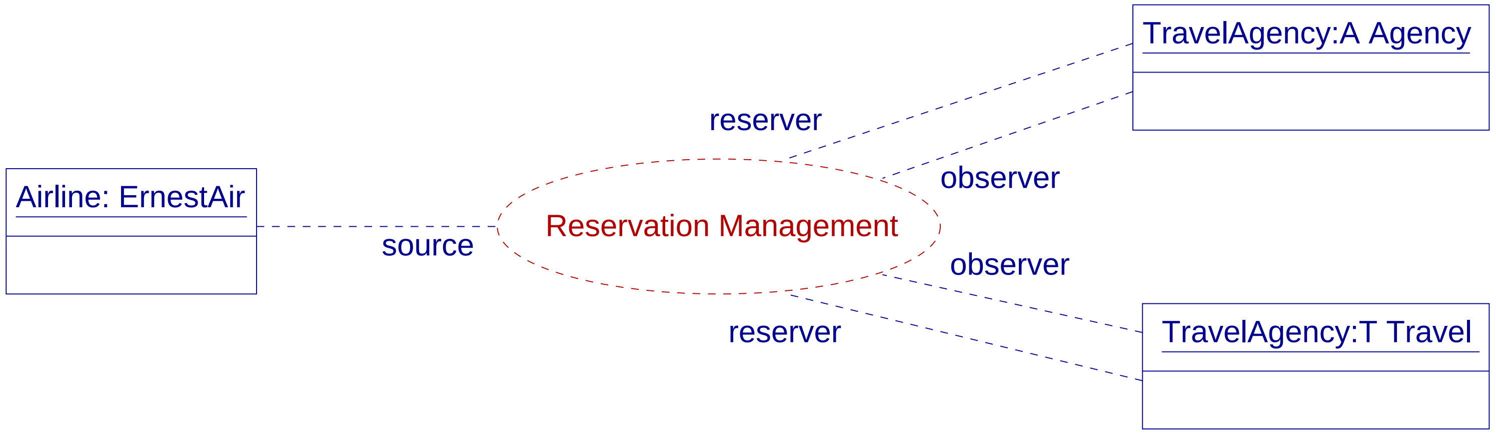
- ➤ Use of generic identifiers
- ➤ Identifier management done in the collaboration

New Car Park Management Application Design



➤ Place set has disappeared ⇒ inside the collaboration

Flight Seat Reservation Application Design



➤ Reuse of the same collaboration ⇒ the same interaction abstraction

INTERACTION ABSTRACTIONS

- ➤ Comparison of the second collaboration design with the first one:
  - ➥ Does “more”, more abstract
  - ➥ Independent, “self-content”, consistent
- ⇒ Define a high-level interaction abstraction:
  - ➥ Easily usable and reusable
  - ➥ At the specification level

INTERACTION ABSTRACTIONS

- ➤ High-level interaction abstractions are useful and interesting in application architecture
  - ➥ At specification level: UML Collaborations are suitable
  - ➥ At implementation level: often, only low-level communication primitives (RPC)

$\Rightarrow$ We propose to use interaction components

  - ➥ For manipulation of high-level interaction abstractions, even at implementation level
  - ➥ An interaction component is specified on the base of a UML collaboration following our specification methodology

PROPERTIES OF SOFTWARE COMPONENTS

- An interaction component is first of all a software component :
  - Independent and deployable software entity
  - Specify offered and required services interfaces
  - Subject to composition with other components

Interaction Components (or Mediums)

Software component integrating any communication (coordination, interaction) system or protocol

- ➤ Independently of its complexity: a consensus protocol, a multimedia stream broadcast, a voting system...
- ➤ At specification level: a UML collaboration following specific design rules
- ➤ At implementation and deployment levels: an instantiable component ⇒ implementation of a UML collaboration

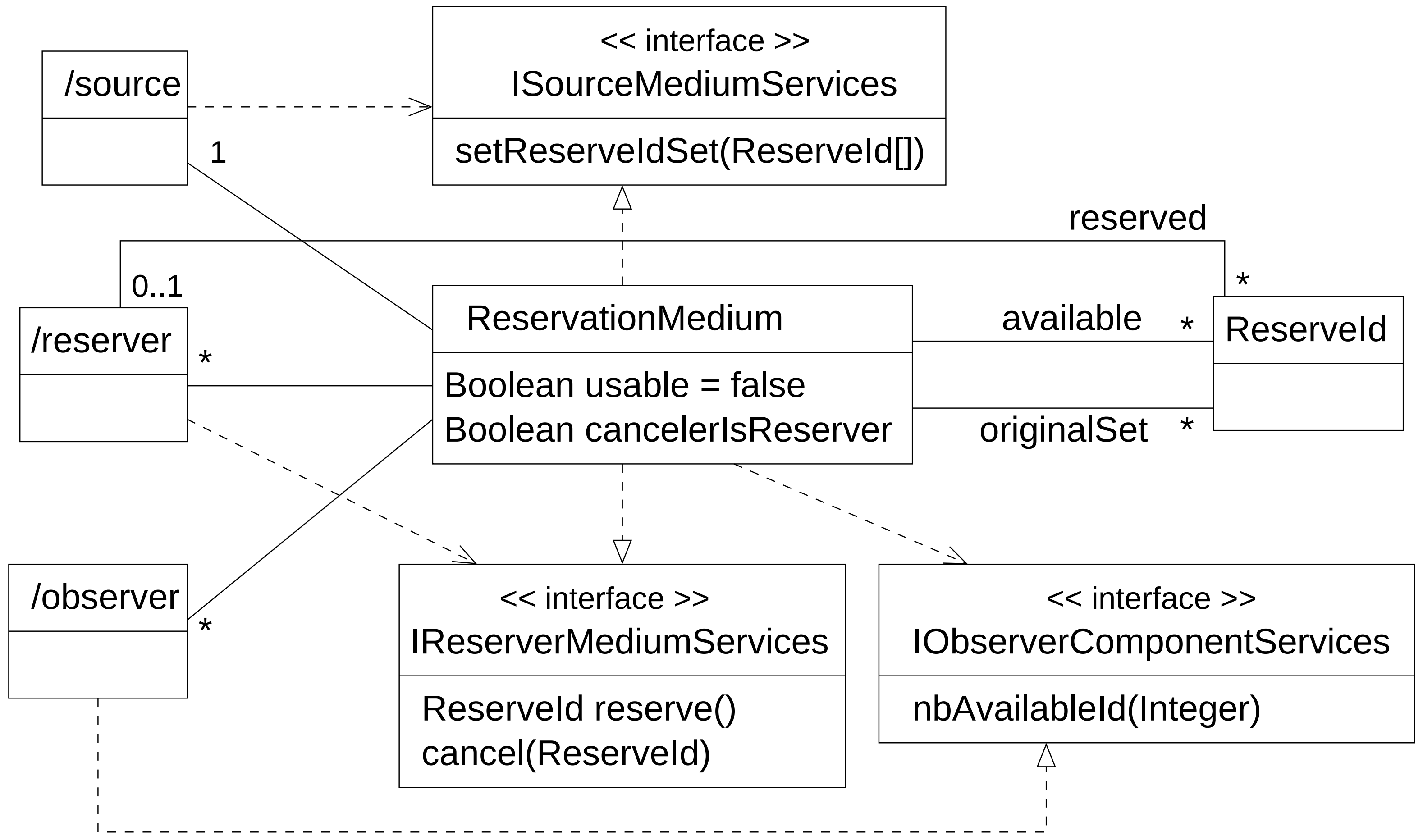
⇒ Reification of an interaction abstraction all along the software process

Specification of a Medium: Usage Contract

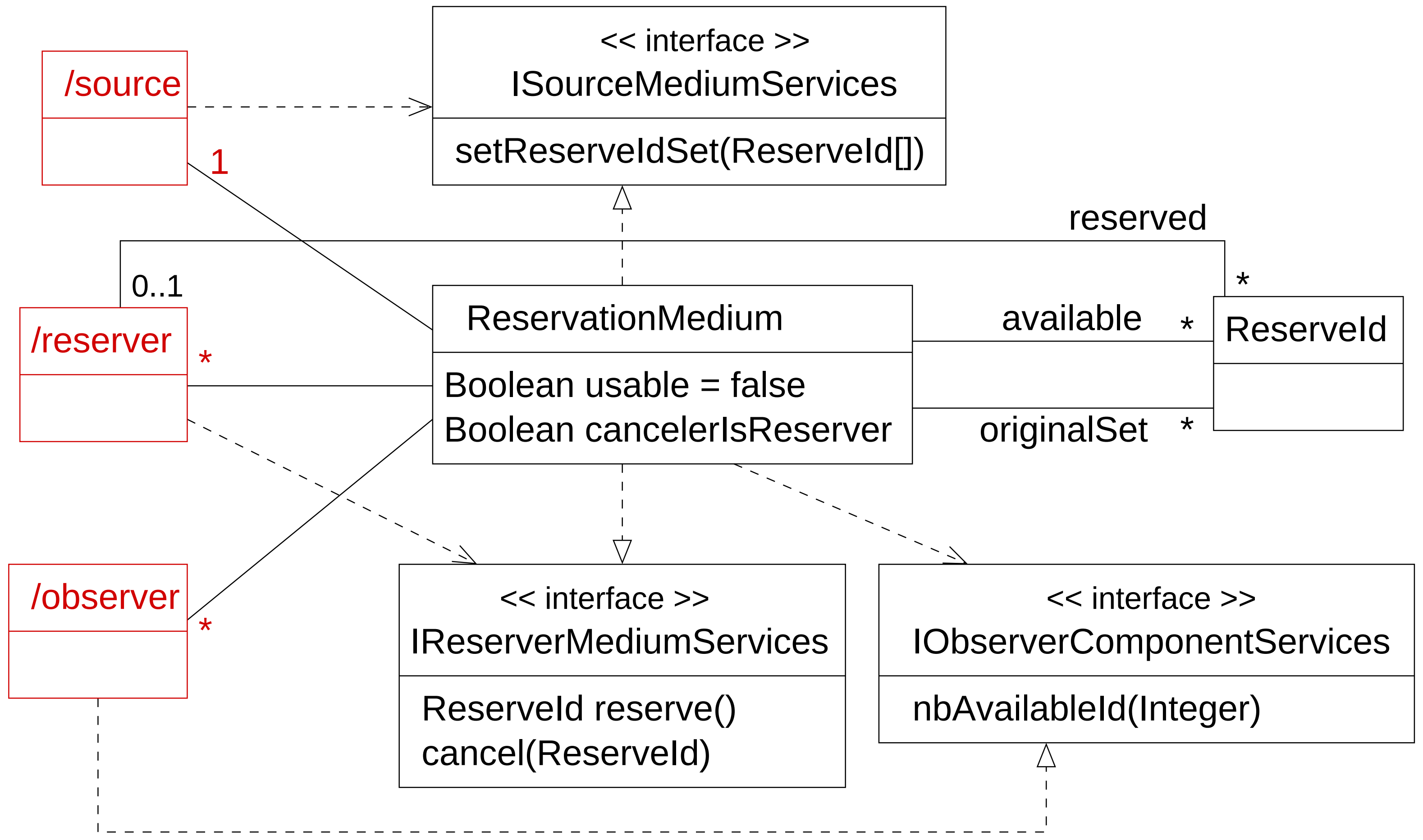
- ➤ On the base of a UML collaboration:
  - ➥ Depending on their needs, components using the medium play different roles
  - ➥ For each role: interfaces of offered and required services
- ➤ OCL for specifying the services semantics
- ➤ Statecharts and messages on collaborations for dynamic behavior
- ➤ Generalization of OCL expressions to link all the views
- ➤ And other UML features if needed

$\Rightarrow$ Abstract specification: without implementation assumption

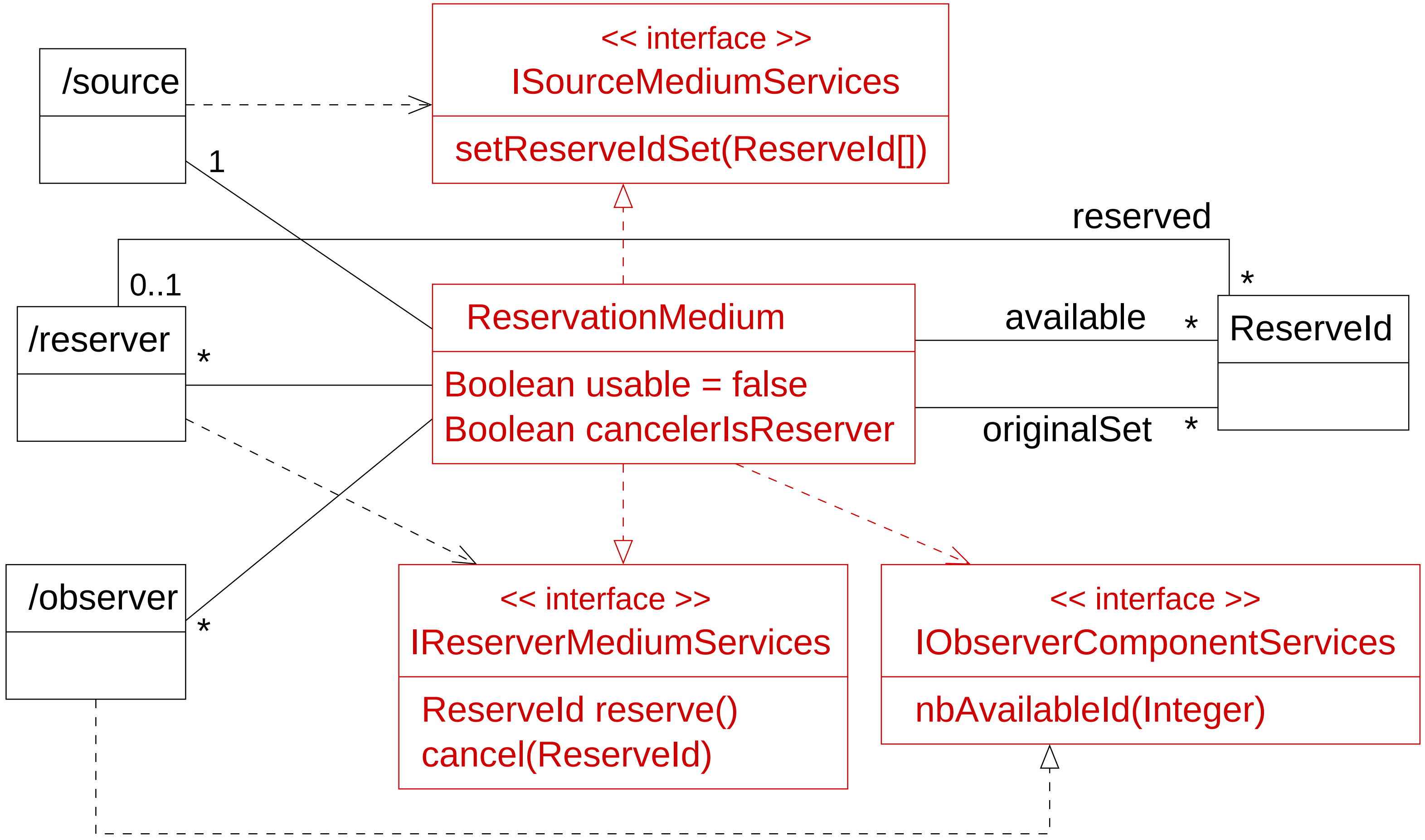
THE RESERVATION MEDIUM



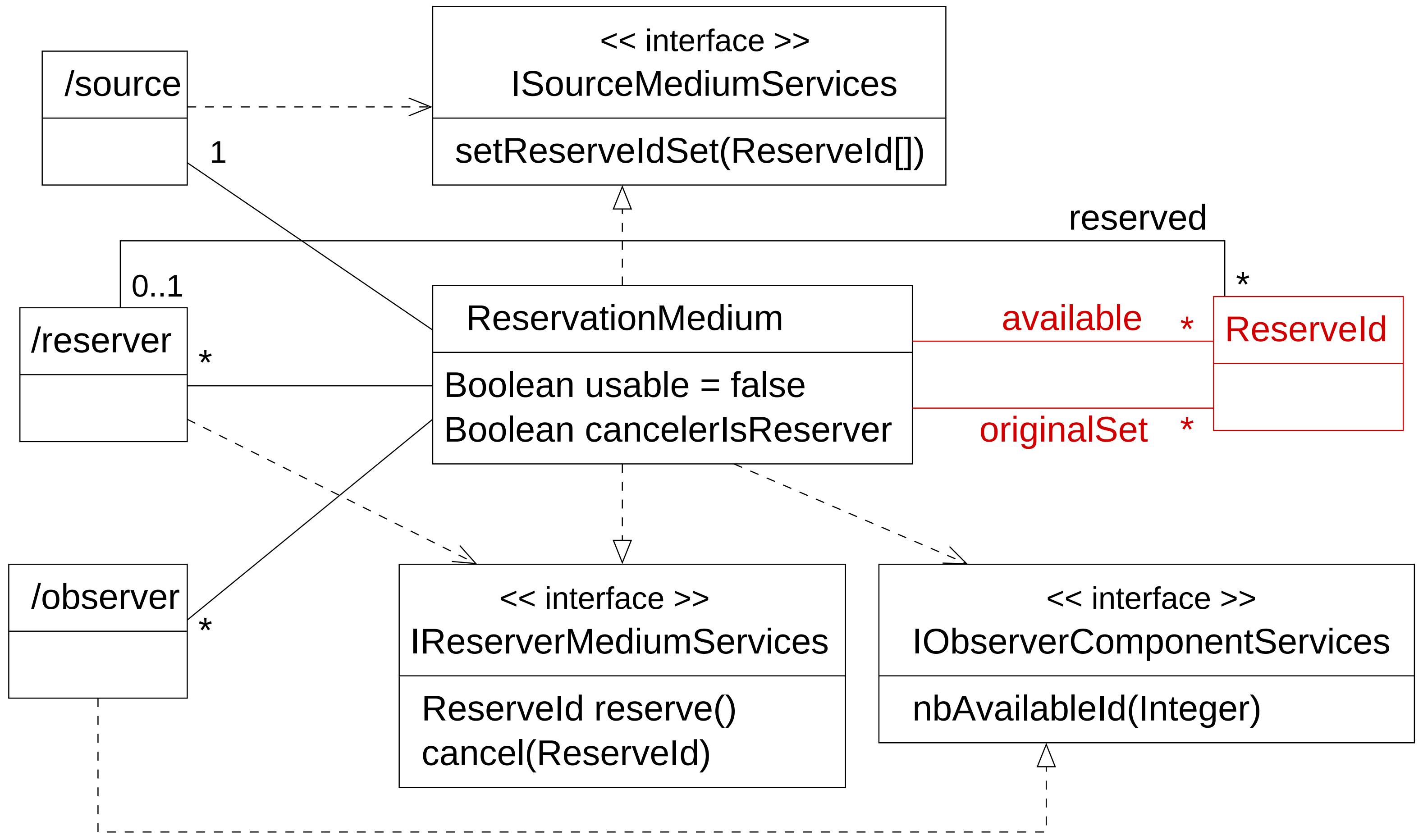
The Reservation Medium



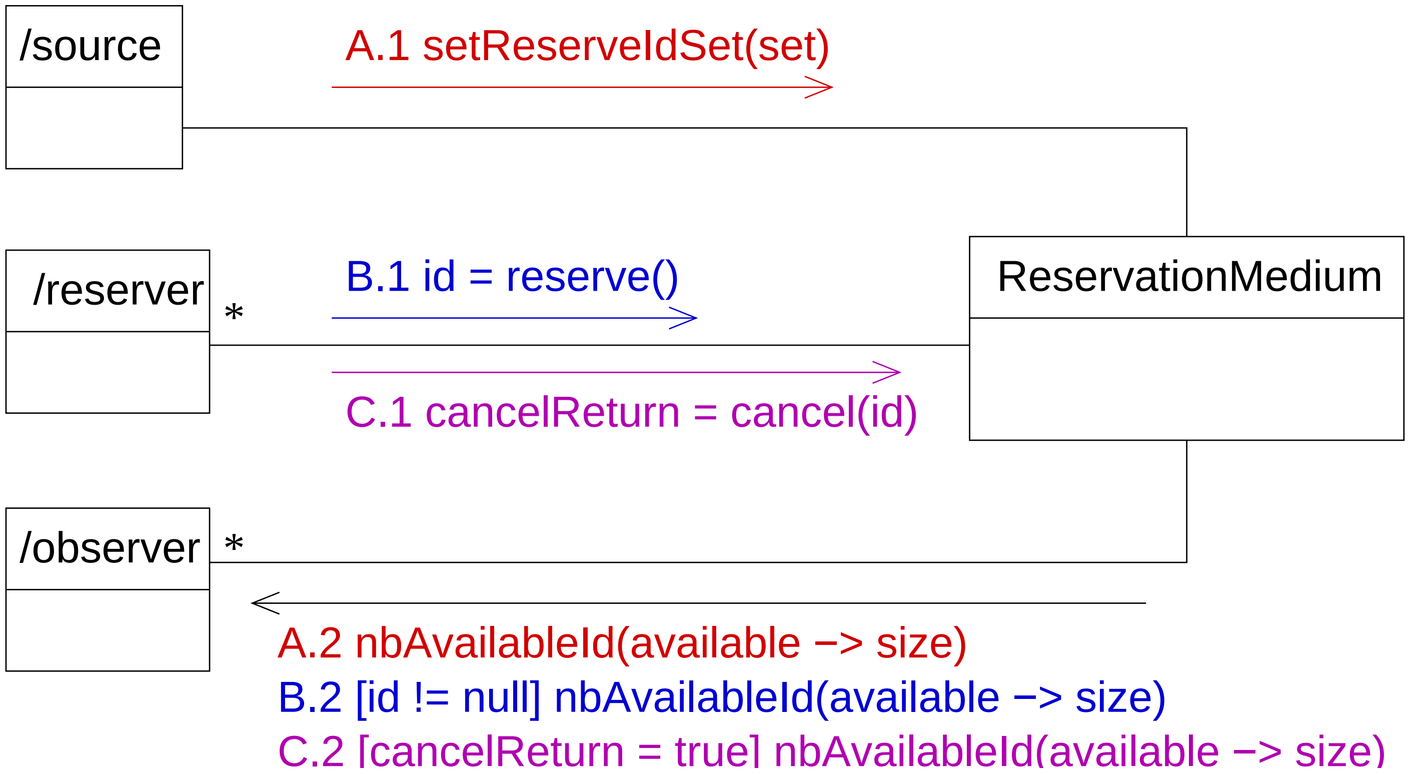
The Reservation Medium



The Reservation Medium



Dynamical View of the Collaboration


/source
A.1 setReserveIdSet(set)
/reserver
*
B.1 id = reserve()
ReservationMedium
C.1 cancelReturn = cancel(id)
/observer
*
A.2 nbAvailableId(available −> size)
B.2 [id != null] nbAvailableId(available −> size)
C.2 [cancelReturn = true] nbAvailableId(available −> size)

OCL CONSTRAINTS FOR SERVICE SEMANTICS

```
context ReservationMedium::
               setReserveIdSet(Set idSet, Boolean cancel)
pre:
     usable = false
post:
     originalSet = idSet
     and available = idSet
     and usable = true
     and cancelerIsReserver = cancel
     and reserver –> forAll( r | r.reserved –> isEmpty )
```

CONCLUSION

Interaction components: reification of interaction abstractions during all the software process

- ➤ Manipulation of high-level interaction abstraction, even at the implementation level
- ➤ Good usability and reusability of interaction abstractions
- ➤ A way to reify and implement a UML Collaboration

CONCLUSION

- Other parts of the work on interaction components:
  - Definition of a specification refinement process: from abstract specification to several implementations
  - A Java framework (downloadable as free software) for implementing mediums and applications in a distributed context
- For more information:
  - Web: http://www-info.enst-bretagne.fr/medium/
  - E-mail: Eric.Cariou@enst-bretagne.fr