EDOC'02 – 19/09/2002

An Architecture and a Process for Implementing Distributed Collaborations

Éric CARIOU, Antoine BEUGNARD, Jean-Marc JÉZÉQUEL

ENST Bretagne

IRISA

INTRODUCTION

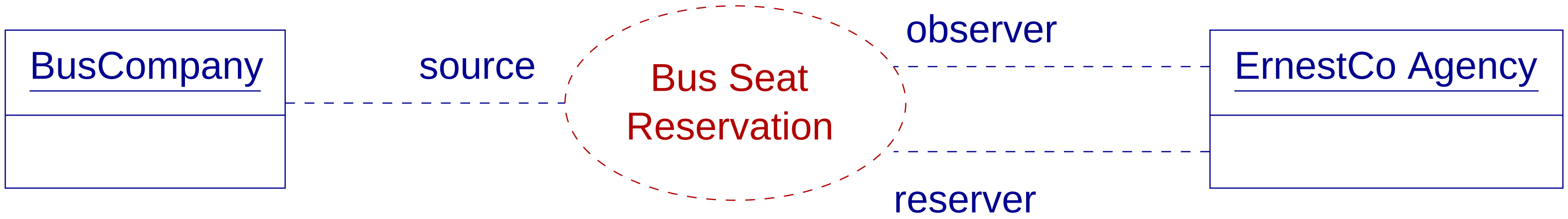
- Key-point in distributed systems: communication among remote components
- Non-functional constraints can impact the implementation
- Our proposition:
  - The reification of interaction abstractions as software components
  - An architecture and a specification process of these components

OUTLINE

1. Study of a reservation system in two different contexts
2. Influence of non-functional constraints
3. Introduction to interaction components
4. How interaction components can help in management of non-functional constraints

Reservation of Places in Buses

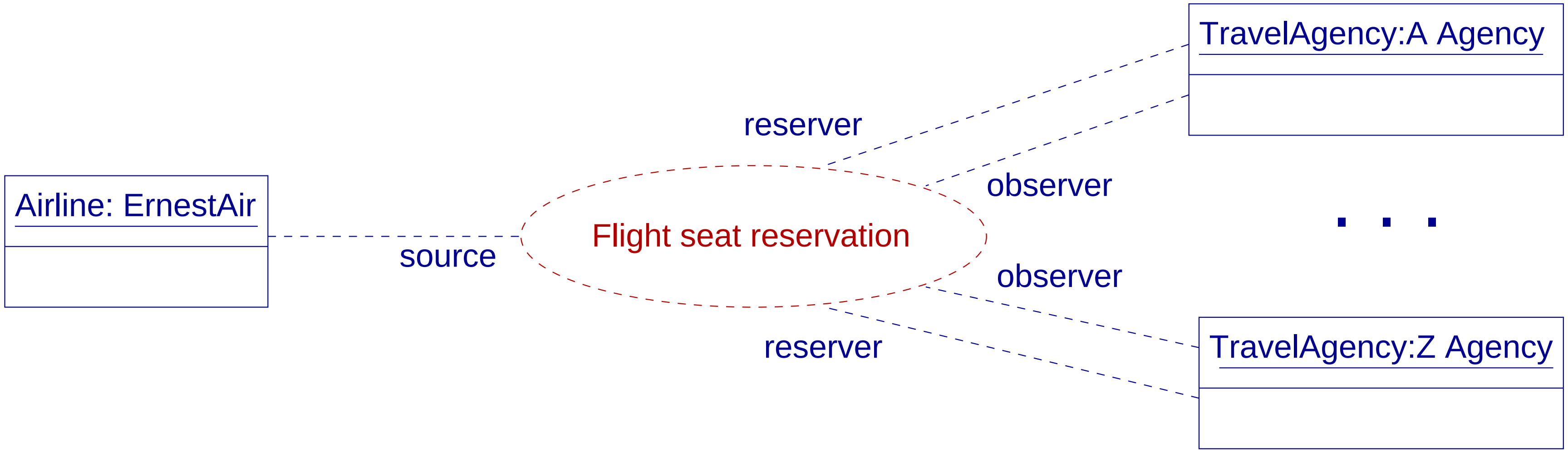
- A small bus company with few journeys
- A single agency sells places in buses for this company



- The components interact through a reservation system

RESERVATION OF PLACES IN FLIGHTS

- ➤ A big airline company with hundreds of flights
- ➤ Thousands of travel agencies worldwide distributed



- ➤ The components interact through a reservation system

AN ABSTRACT RESERVATION SYSTEM

- In both applications:
  - Reservers components: reservation of identifiers (places) and cancellation of reservations
  - Source components: addition and removal of informations on resources (buses or planes)
- Same requirements $\Rightarrow$ same reservation abstraction, same interaction abstraction

THE RESERVATION SYSTEM IMPLEMENTATION

- But the context is different:
  - Number and localization of interacting components
  - Number of data to handle
- A single small data server is enough for the first case but not for the second $\Rightarrow$ need different implementations to face scalability

$\Rightarrow$ same functional requirements but different implementations

INTERACTION ABSTRACTIONS

- Non-functional constraints (e.g. scalability, security, reliability) impact the implementation of an interaction abstraction
- Some questions:
  - How to specify an interaction abstraction ?
  - How to have several implementations of the same abstraction ?

⇒ we propose to use interaction components

Interaction components (or mediums)

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

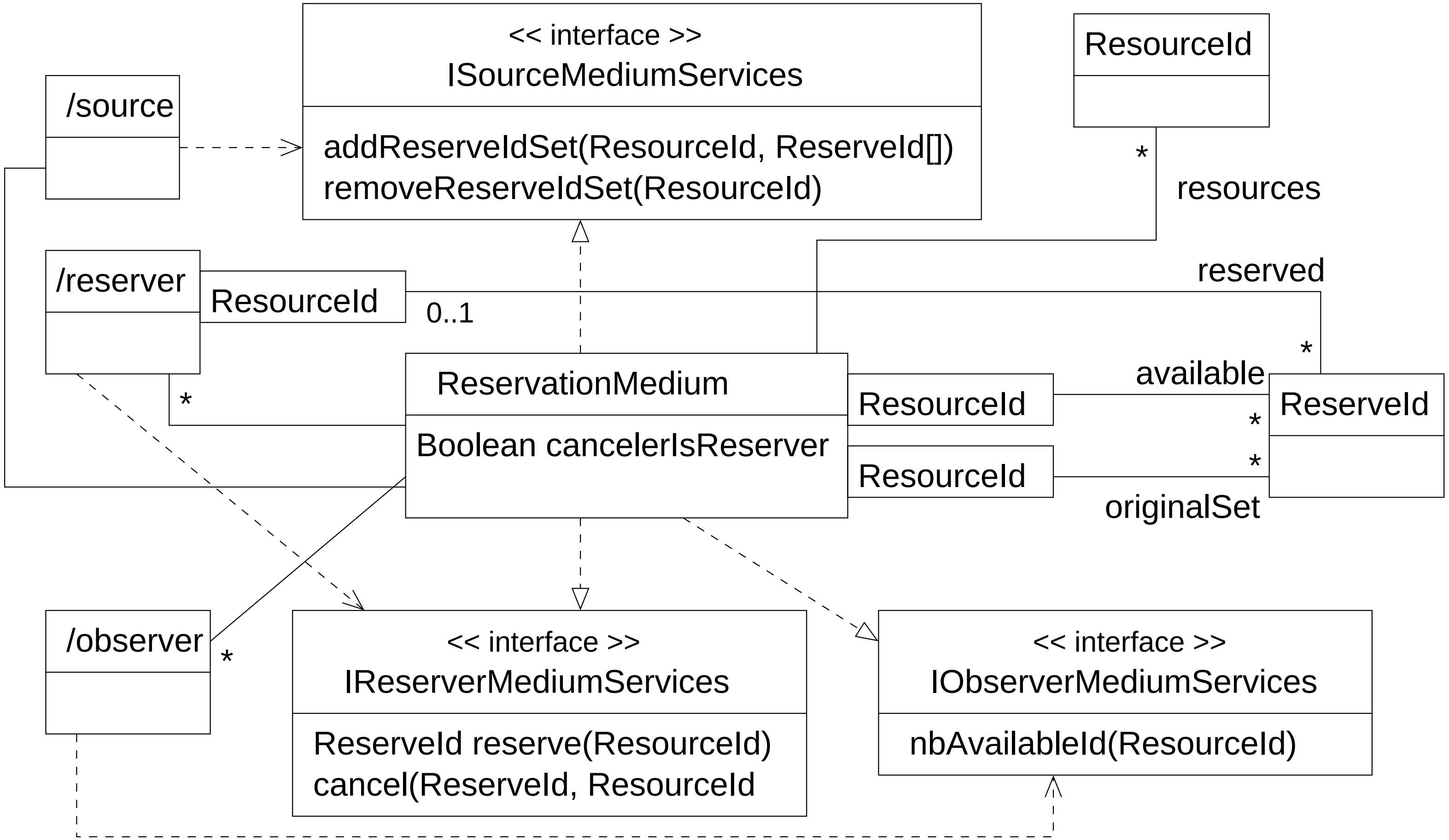
- ➤ Independently of its complexity: a consensus protocol, a multimedia stream broadcast, a voting system...
- ➤ At specification level: a UML collaboration following specific design rules
- ➤ At implementation and deployment levels: an instantiable component

$\Rightarrow$ reification of an interaction abstraction during all the software process

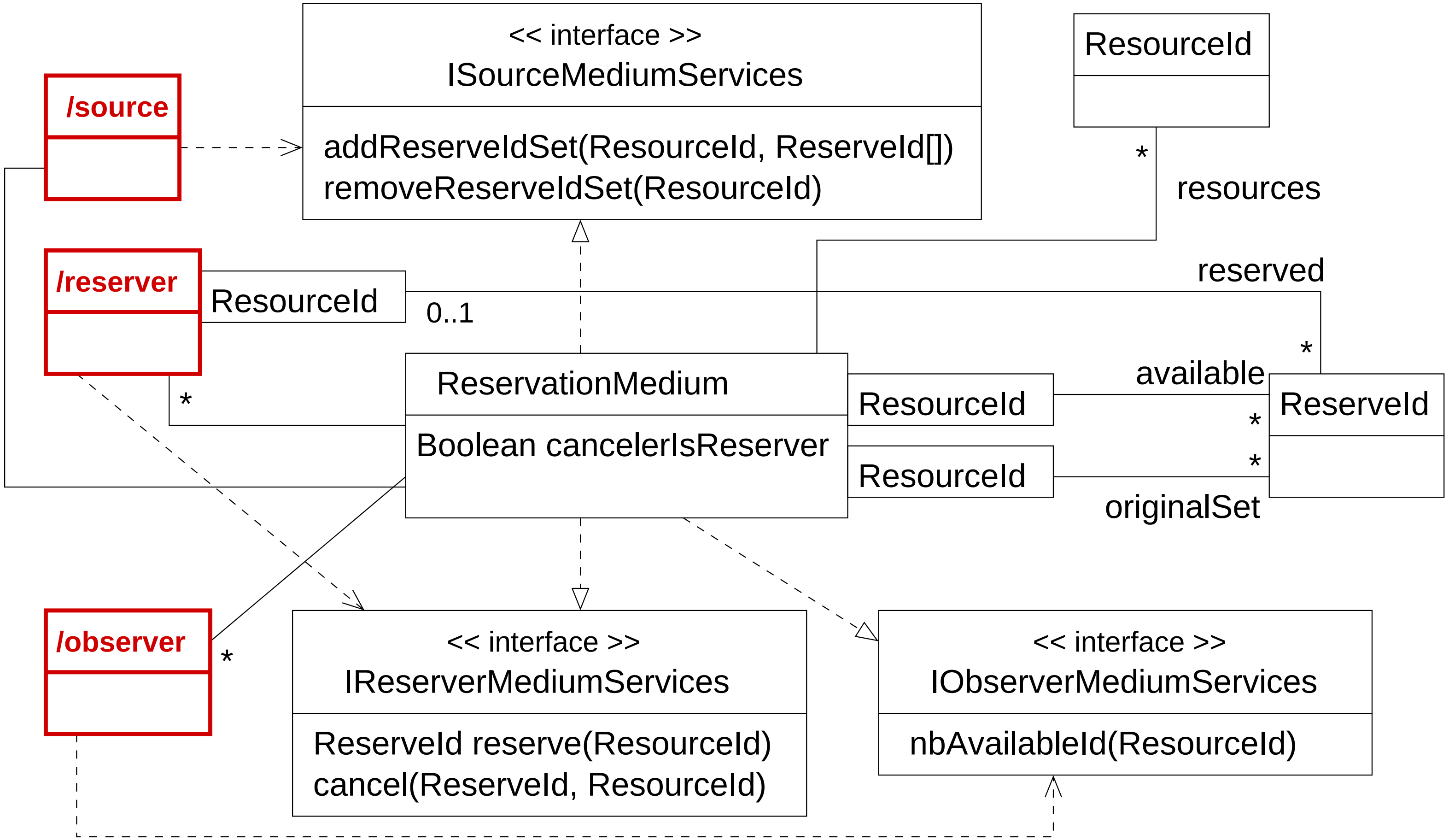
Specification of a medium: usage contract

- ➤ A UML collaboration specifies a medium
  - ➥ Depending on their needs, components using the medium play different roles
  - ➥ For each role: interfaces of offered and required services
- ➤ OCL and others UML features for specifying the services semantics
- ➤ Abstract specification: without implementation assumption

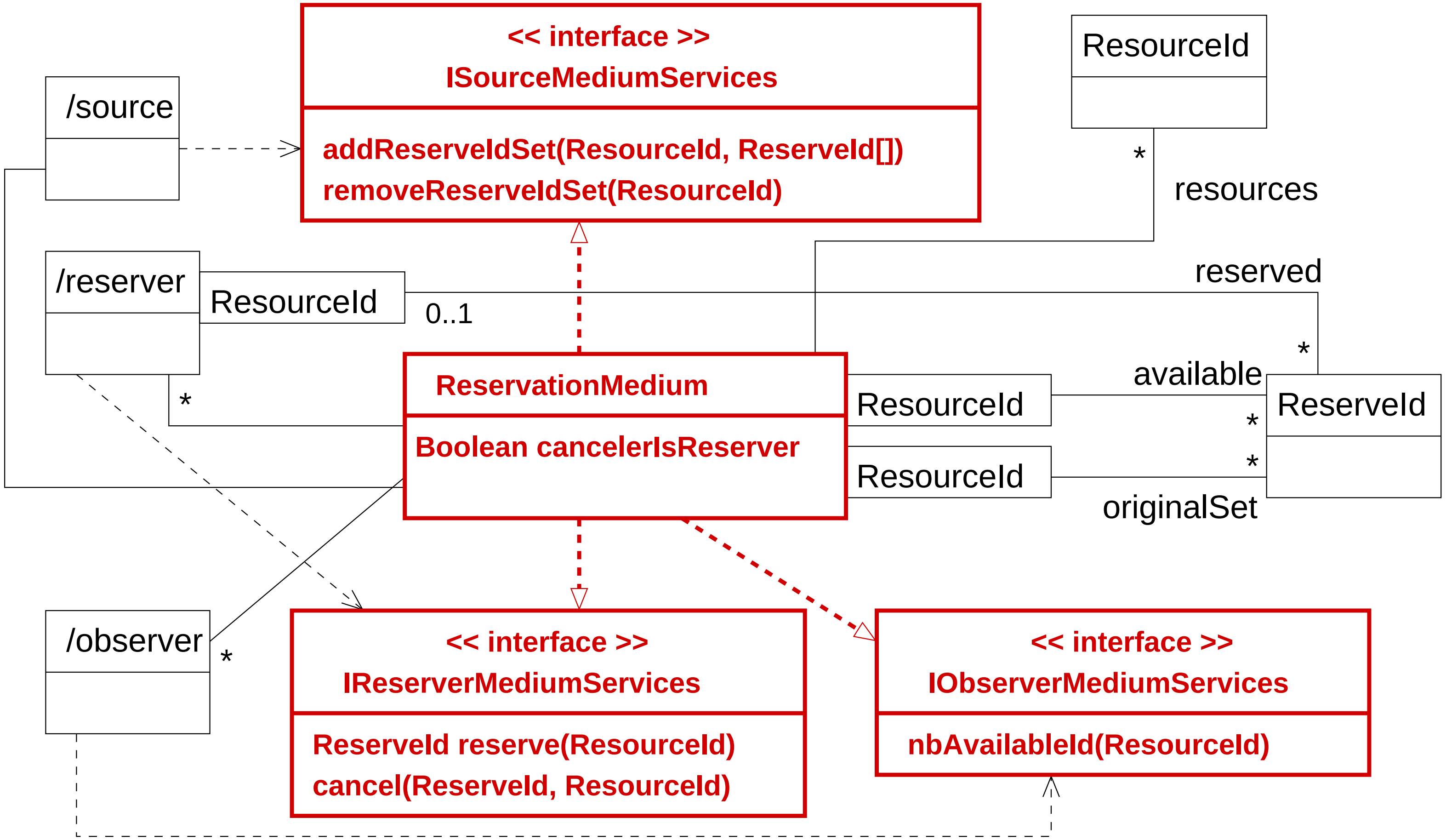
THE RESERVATION MEDIUM



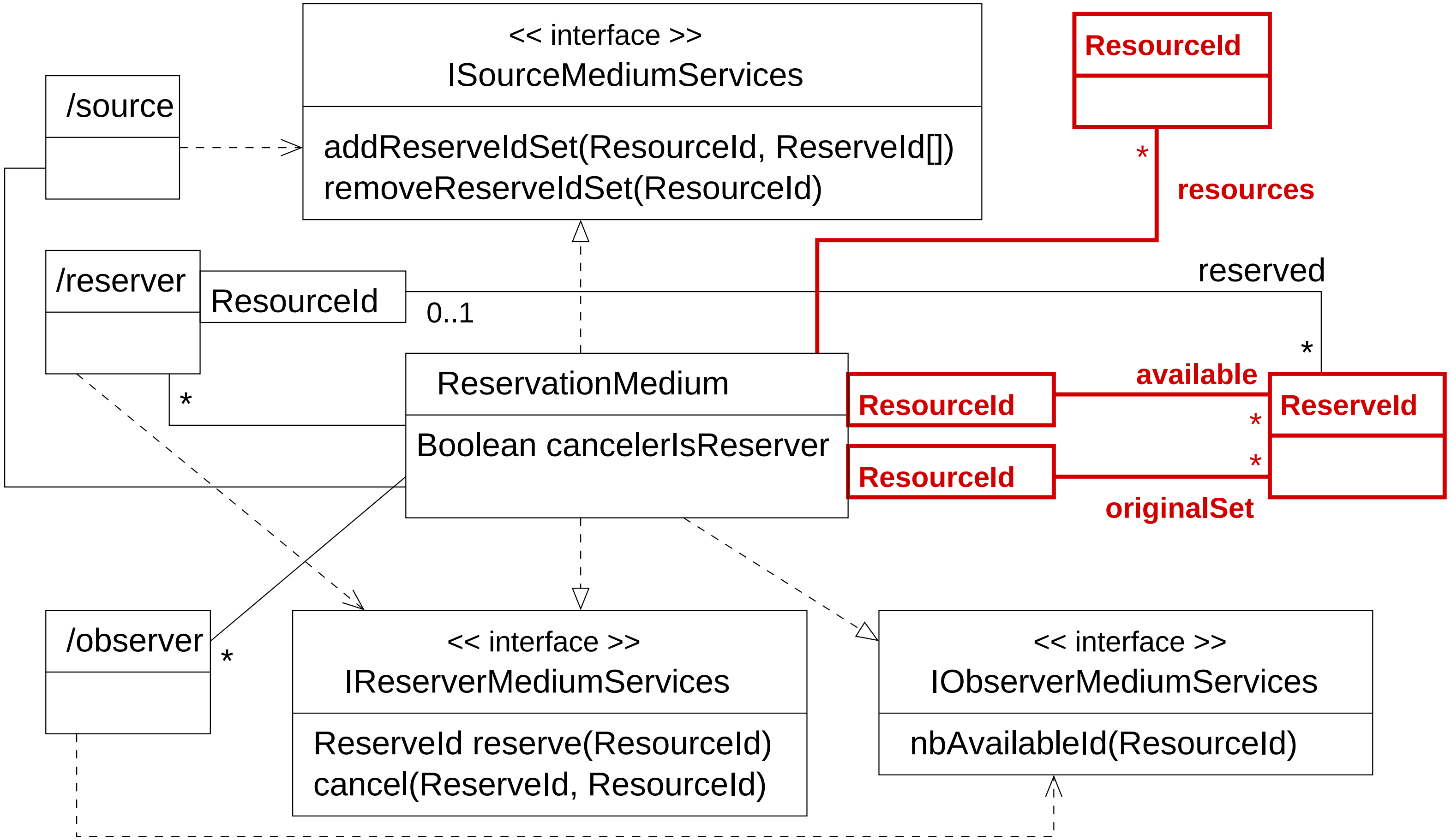
THE RESERVATION MEDIUM



THE RESERVATION MEDIUM

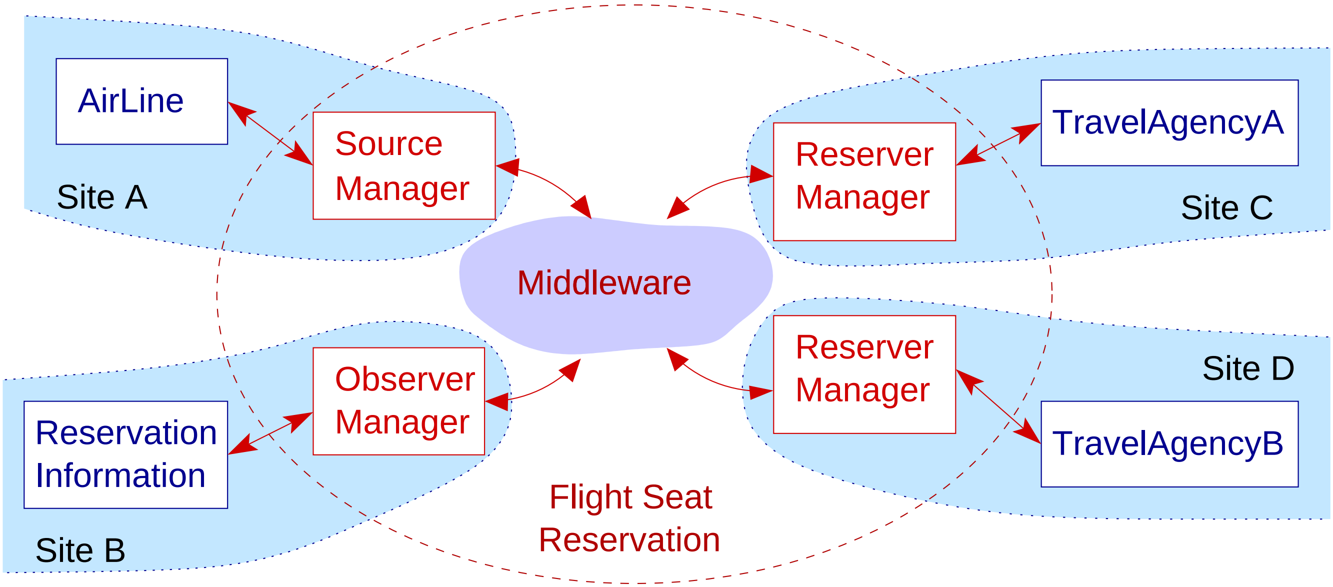


THE RESERVATION MEDIUM



DEPLOYMENT ARCHITECTURE OF A MEDIUM

- A “role manager” is locally associated with each component
- Medium = logical unit composed of all the role managers
- A role manager can be as complex as required



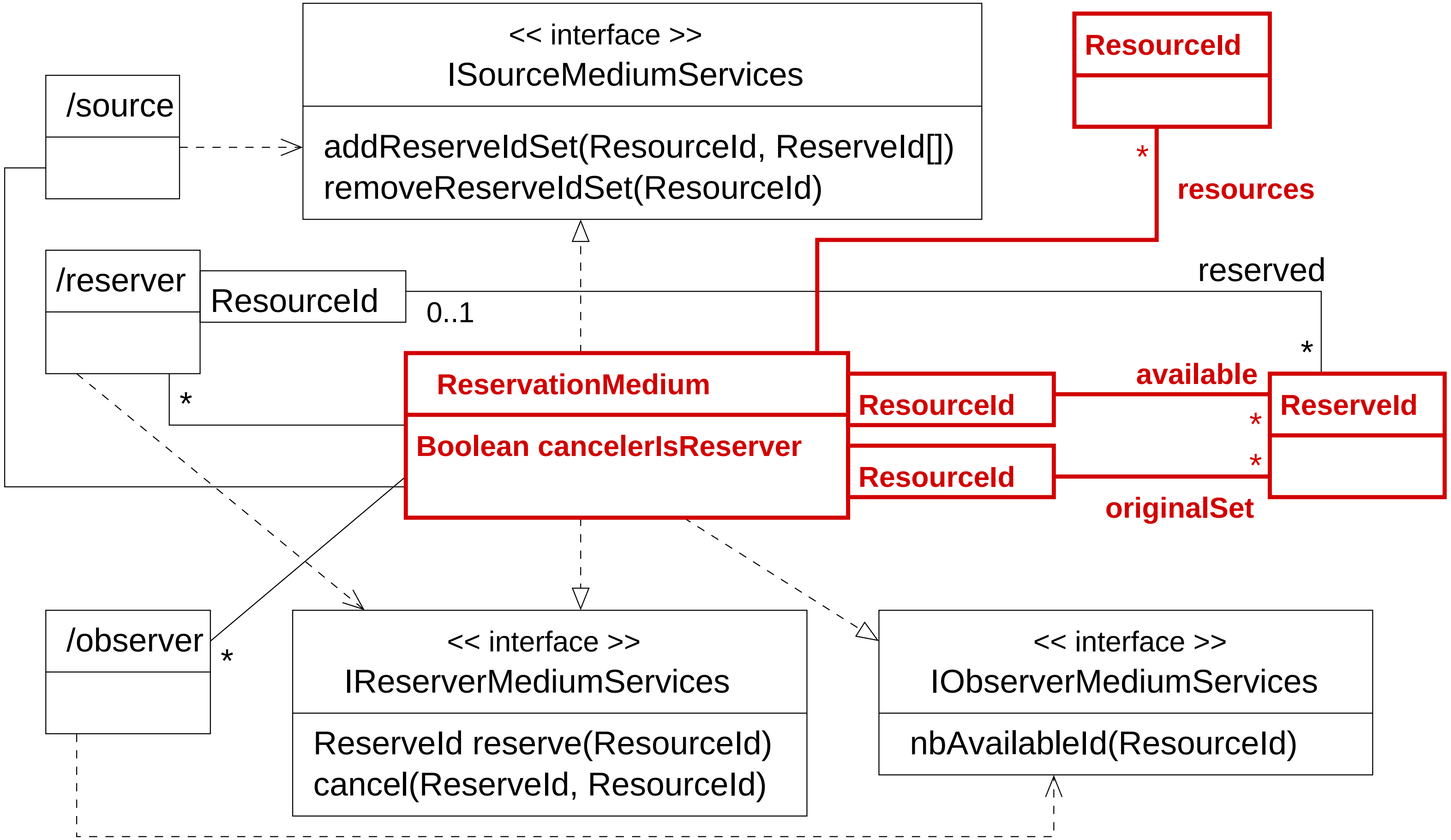
Advantages of this architecture

- ➤ Several implementations of the same abstraction are easily realizable
- ➤ Good separation of functional and interactional concerns even at implementation level

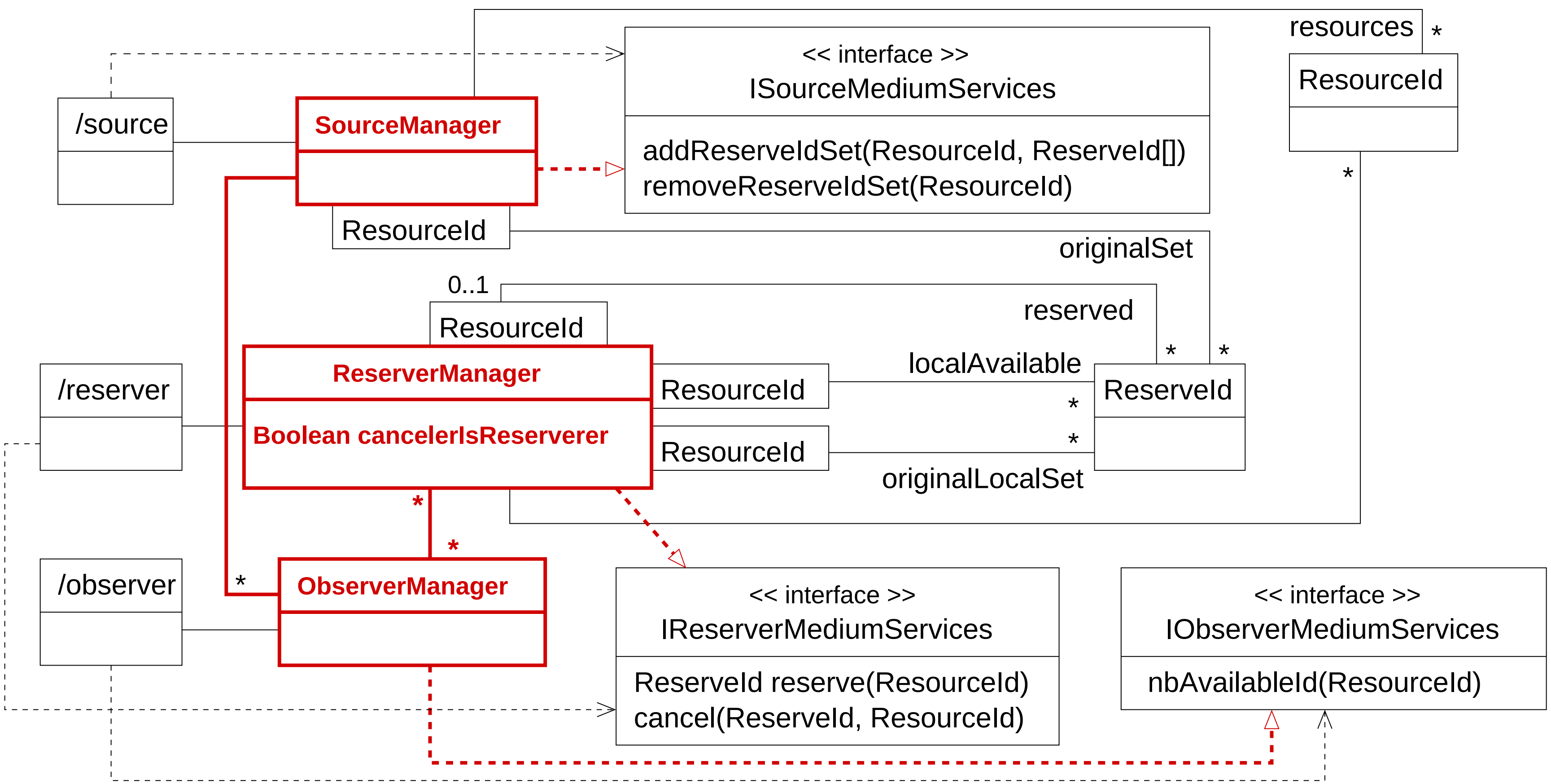
FROM ABSTRACT SPECIFICATION TO IMPLEMENTATIONS

- Specification refinement process:
  - Transform an abstract specification into an implementation one according to implementation choices or constraints
  - Transform the single UML class medium into a set of role managers classes to match the deployment architecture
  - From usage contract to implementation contract
- A single abstract specification can lead to several implementation designs

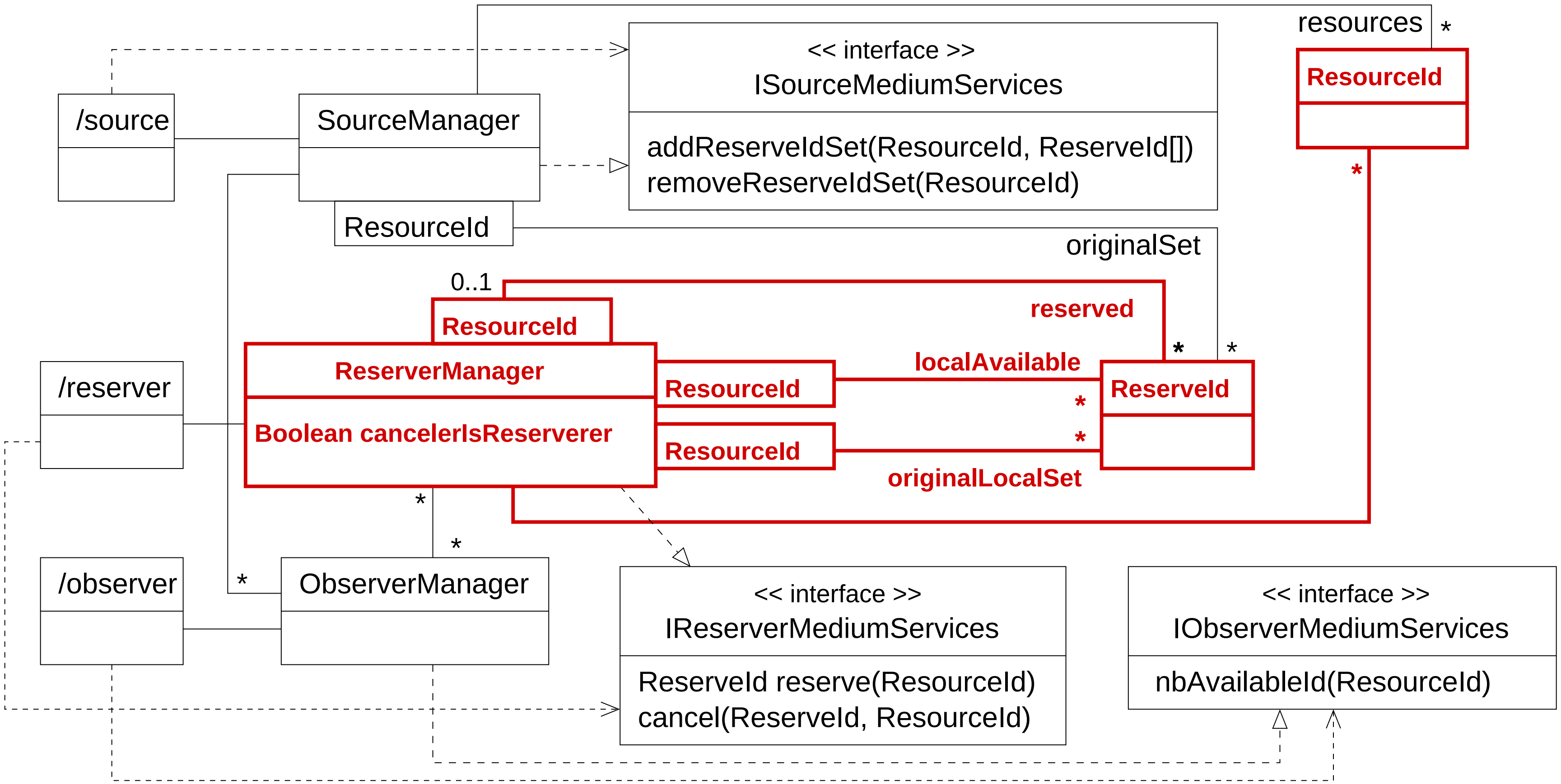
The reservation medium at abstract level



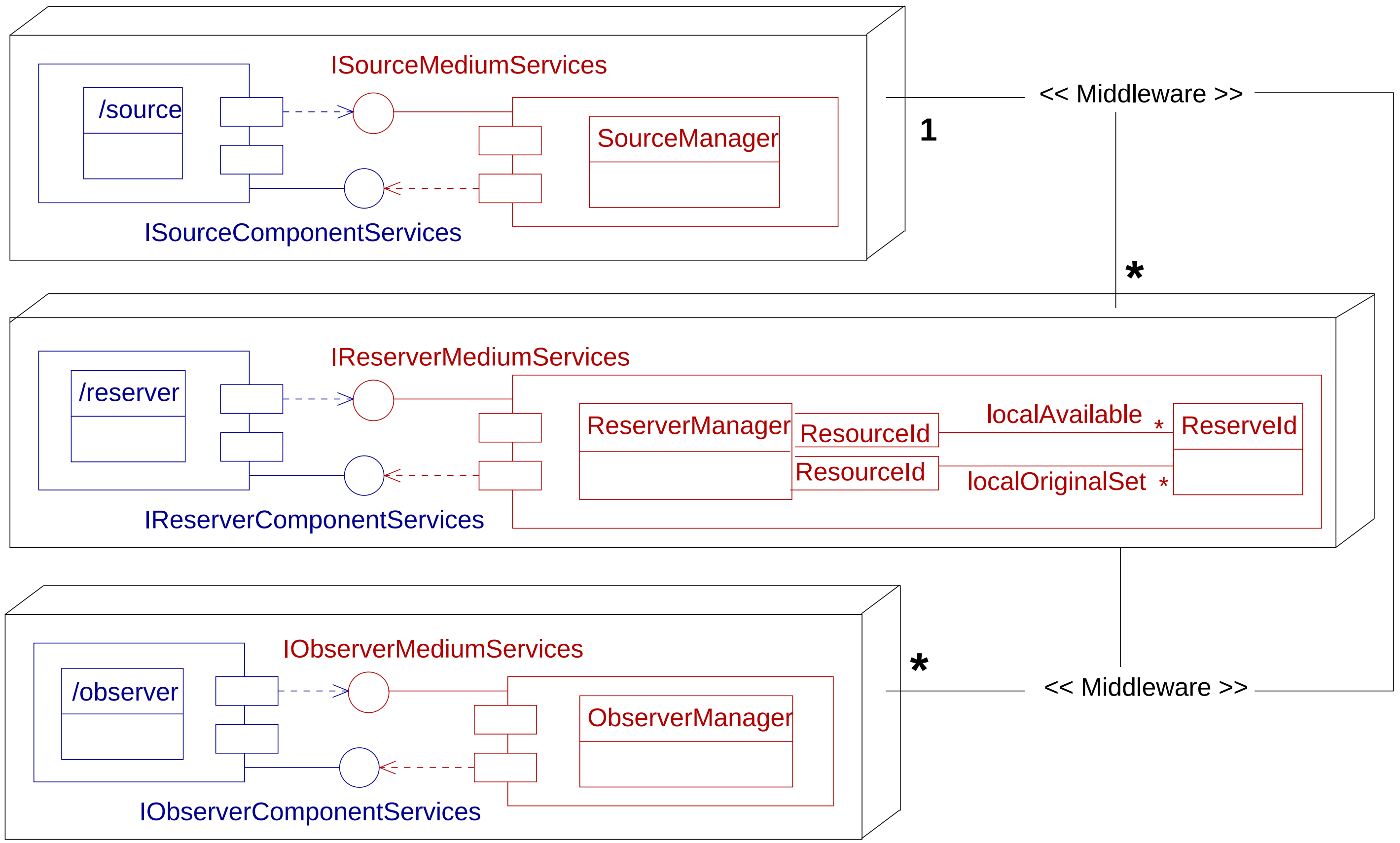
Example: distributed data management choice



EXAMPLE: DISTRIBUTED DATA MANAGEMENT CHOICE



DEPLOYMENT VIEW



Conclusion

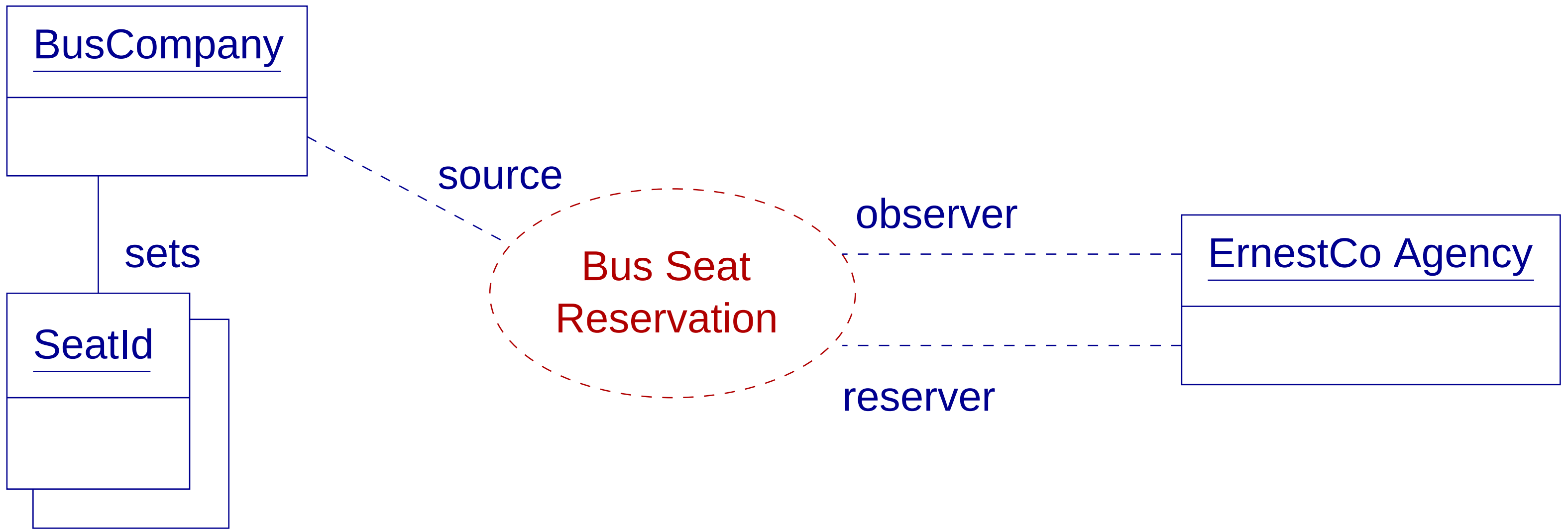
Interaction component: reification of interaction abstraction during all the software process

- Advantages for the interaction management:
  - Good separation of functional and interactional concerns even at the implementation and deployment levels
  - Good reusability of interaction abstractions
- A deployment architecture and a refinement process:
  - From abstract specification to several implementations
  - Selection of the adapted implementation depending on the context or non-functional constraints (e.g. scalability)

CONCLUSION

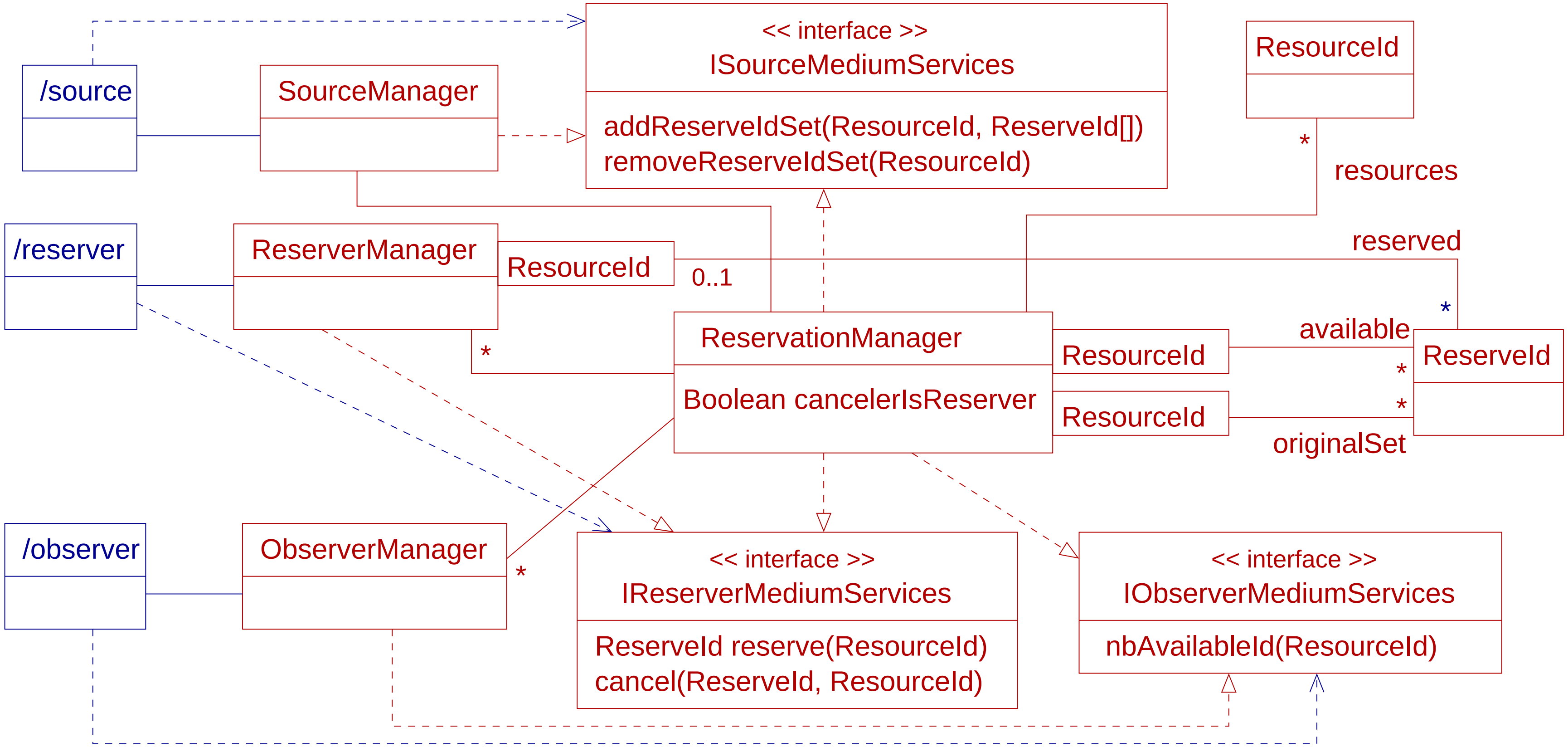
- A Java framework for implementing mediums:
  - Easy use of interactions components in applications
  - Easy implementation of different version of a same abstraction
  - Downloadable as free software (GPL licence)
- For more information:
  - Web: http://www-info.enst-bretagne.fr/medium/
  - E-mail: Eric.Cariou@enst-bretagne.fr

BAD DESIGN FOR THE RESERVATION INTERACTION



- Identifiers managed outside the collaboration
- An implementation choice is already done ⇒ less implementation variants are available

Centralized data management



CENTRALIZED DATA MANAGEMENT