Satisfying requirements for pervasive service compositions

Luca Cavallaro, Pete Sawyer, **Daniel Sykes**,
Nelly Bencomo, Valérie Issarny

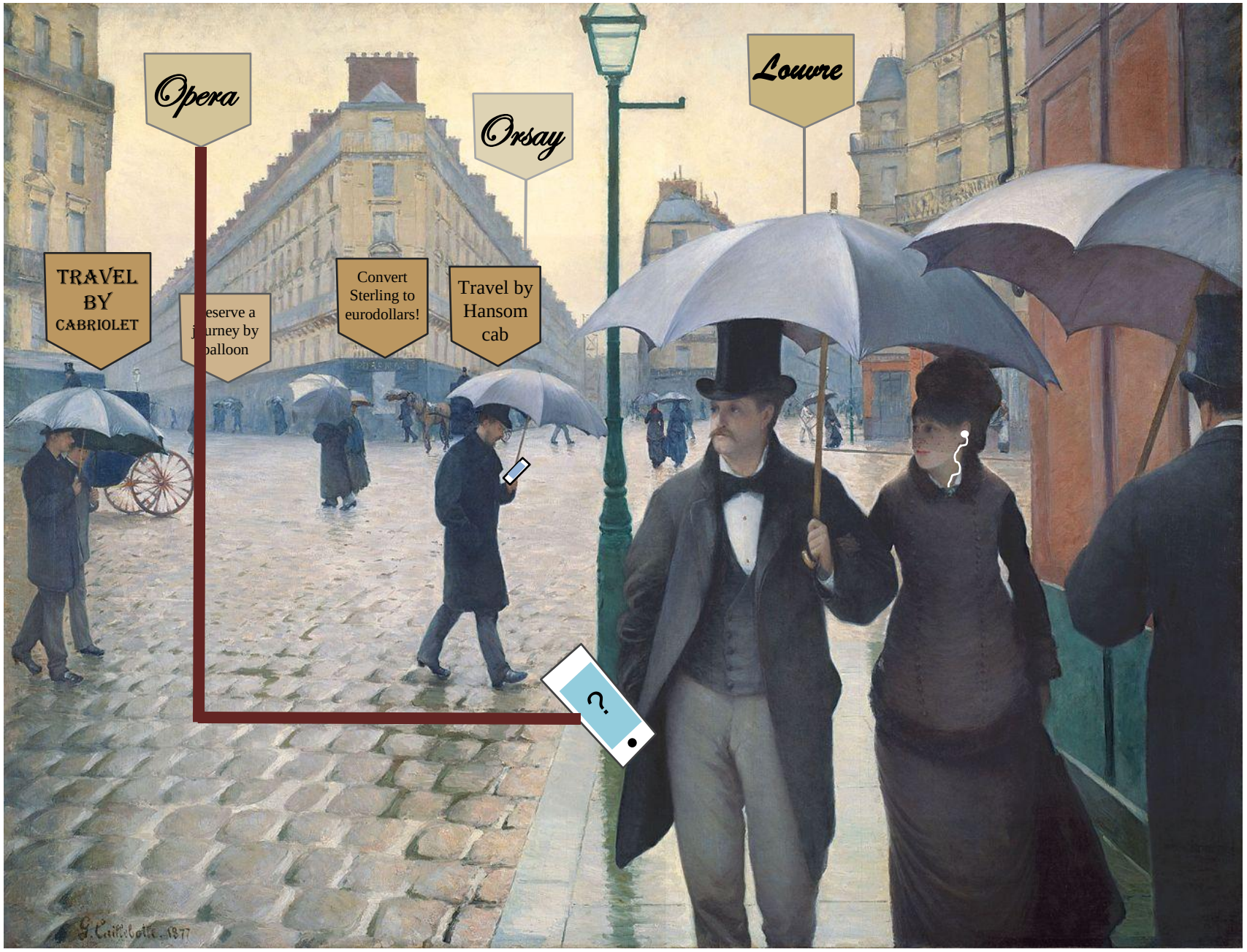
Lero, Lancaster, INRIA

Models @ Runtime, 2nd October 2012

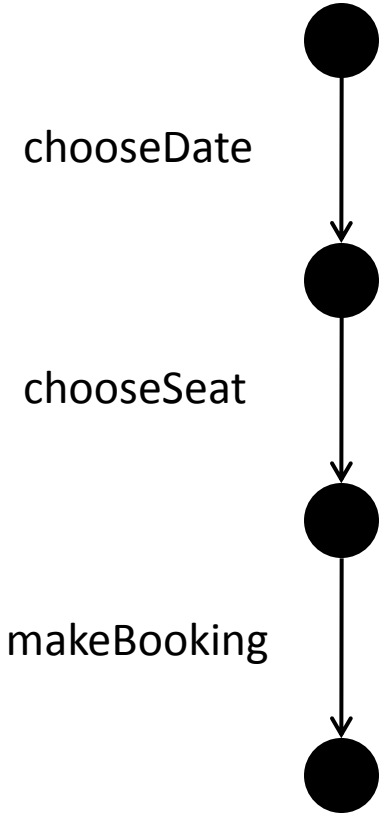
Pervasive environment

- Cannot predict at design time what services will be available
- Services and devices appearing and disappearing all the time
- Huge variety of platforms, protocols, standards and functionality
- How do we **compose** services at **runtime** to achieve our aims?

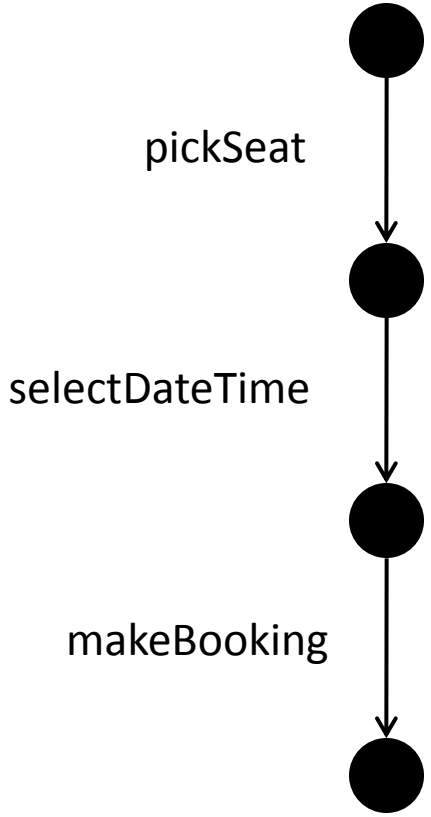

Opera
Orsay
Louvre
TRAVEL BY CABRIOLET
Reserve a journey by balloon
Convert Sterling to eurodollars!
Travel by Hansom cab
?


Opera
Orsay
Louvre
TRAVEL BY CABRIOLET
eserve a
urney by
balloon
Convert Sterling to eurodollars!
Travel by Hansom cab
?

Ticket booking app
(on smartphone)

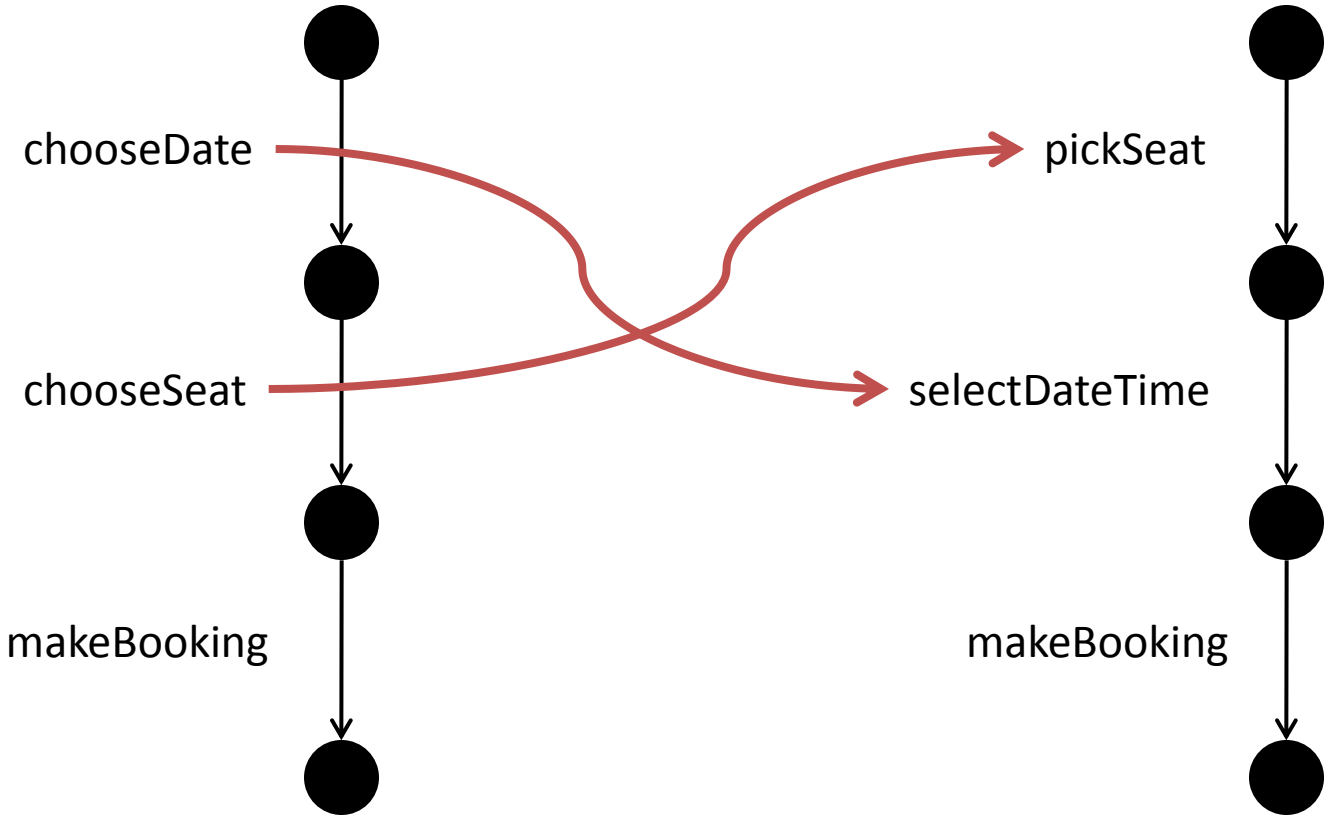


Ticket booking service
(on server)



Ticket booking app
(on smartphone)

Ticket booking service
(on server)



Incompatible: signature & protocol mismatch

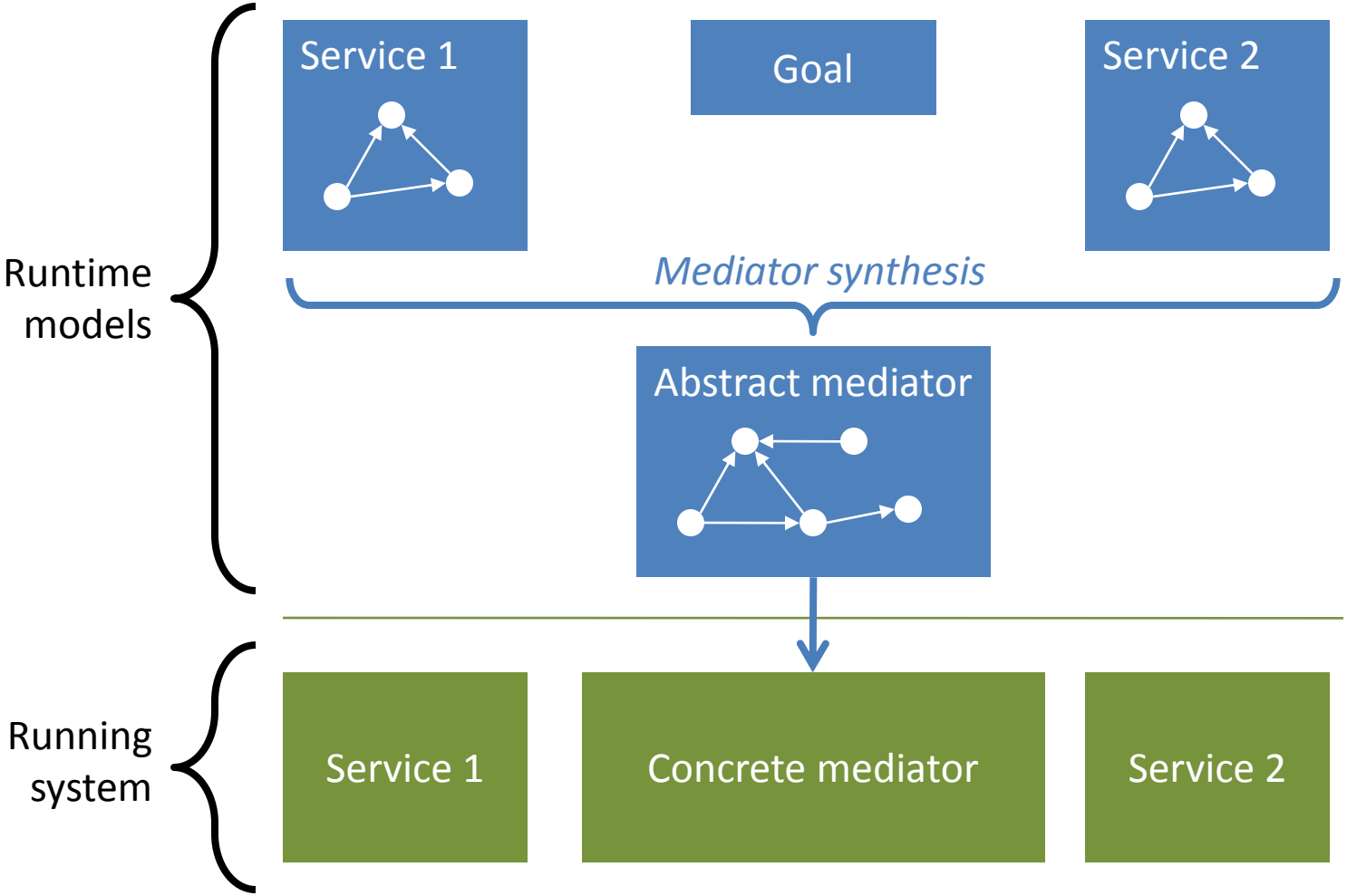
Possible solutions

- Standardisation of interfaces
  - So many standards
  - Often little incentive to standardise
- Manual translation between interfaces
  - Costly, slow
  - Cannot be applied at runtime

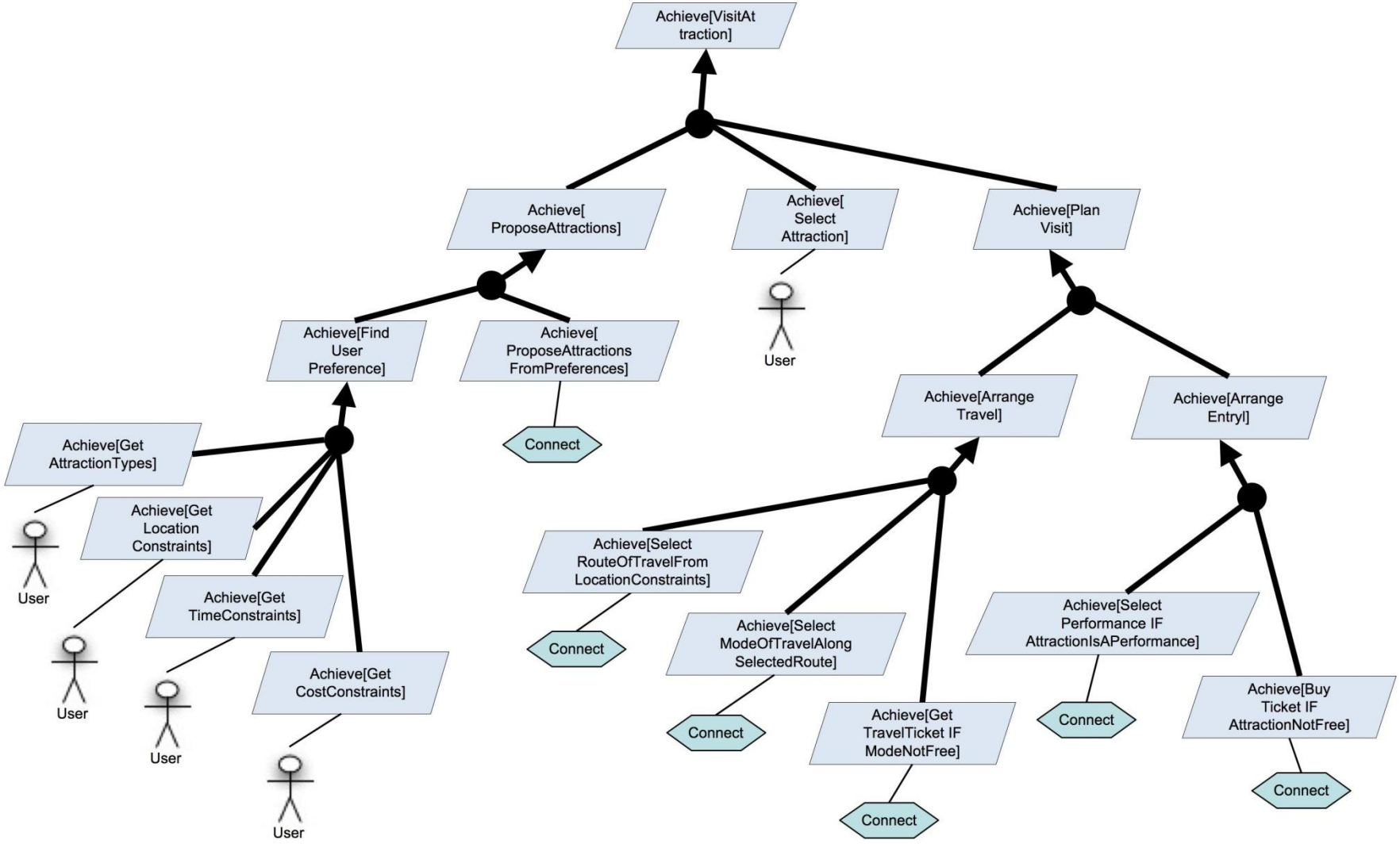
Runtime pervasive composition

- **Discover** services at runtime (WS-Discovery, UPNP)
- **Select** services relevant to **goal**
- Analyse their descriptions and **synthesise** a **mediator**
- Compose mediator with services to satisfy goal
- Adapt by replacing services that disappear

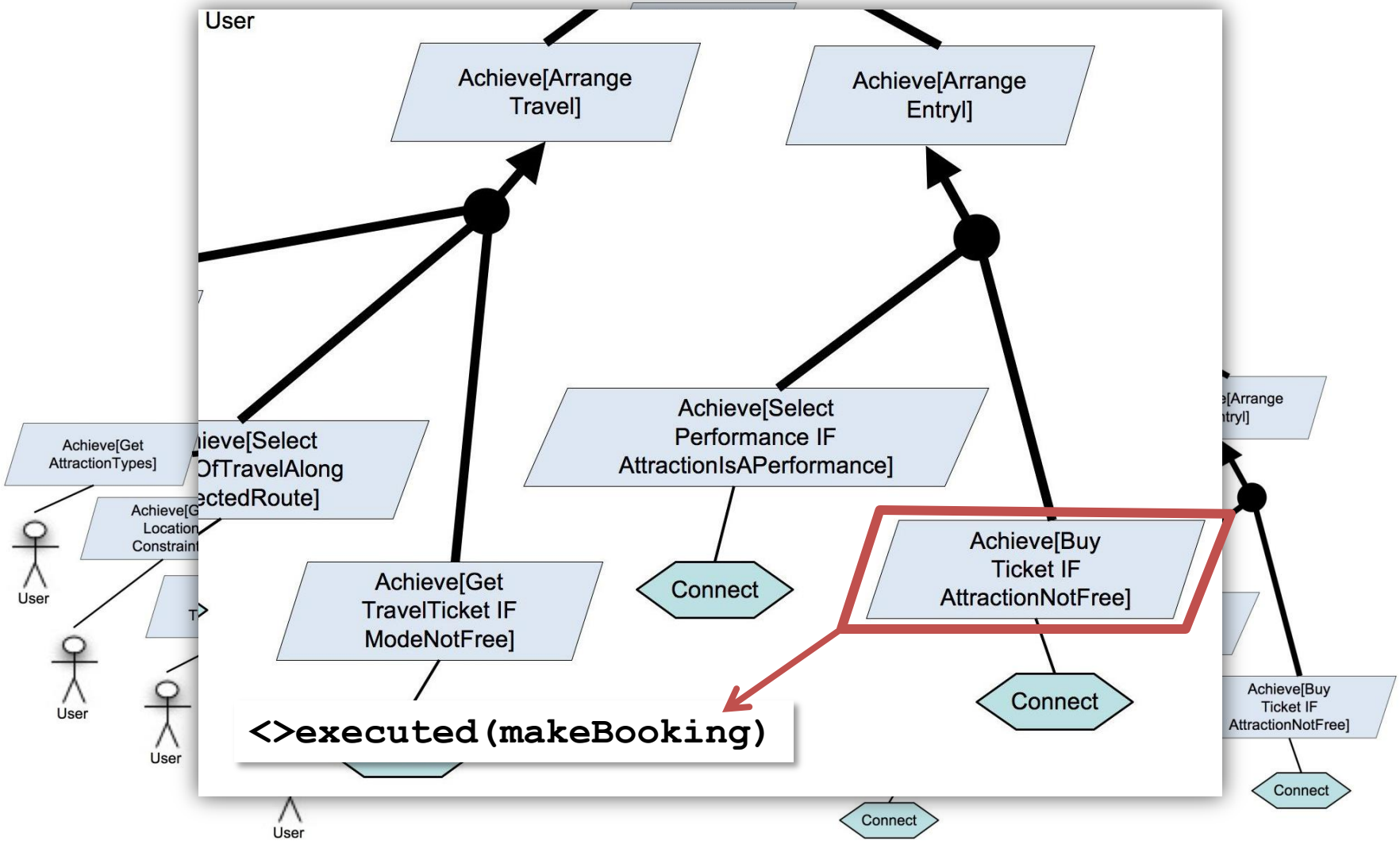
Runtime pervasive composition



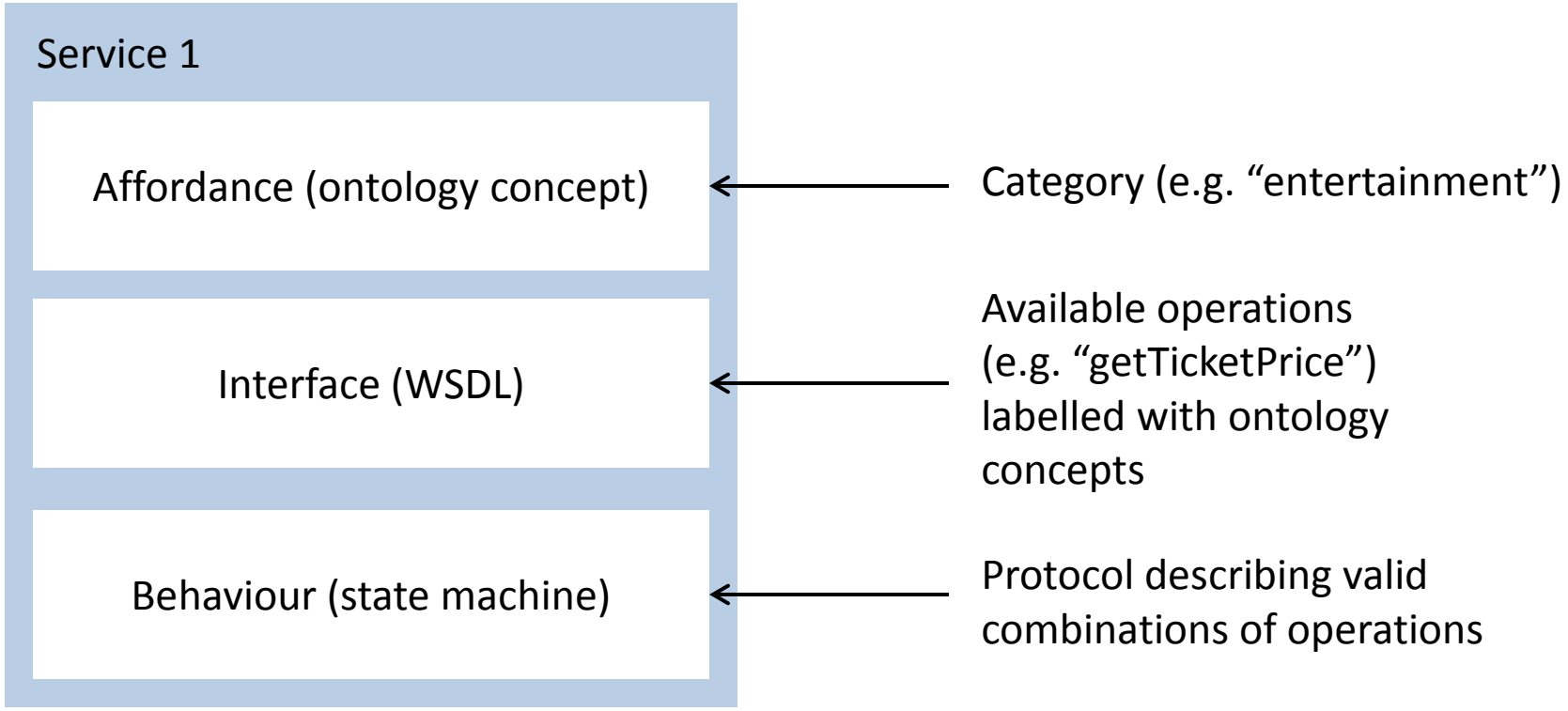
Goal model


Achieve[VisitAttraction]
Achieve[ProposeAttractions]
Achieve[Select Attraction]
Achieve[Plan Visit]
Achieve[Find User Preference]
Achieve[ProposeAttractions FromPreferences]
User
Achieve[Arrange Travel]
Achieve[Arrange Entryl]
Achieve[Get AttractionTypes]
Connect
Achieve[Get Location Constraints]
Achieve[Get TimeConstraints]
Achieve[Get CostConstraints]
Achieve[Select RouteOfTravelFrom LocationConstraints]
Achieve[Select ModeOfTravelAlong SelectedRoute]
Achieve[Get TravelTicket IF ModeNotFree]
Achieve[Select Performance IF AttractionIsAPerformance]
Achieve[Buy Ticket IF AttractionNotFree]

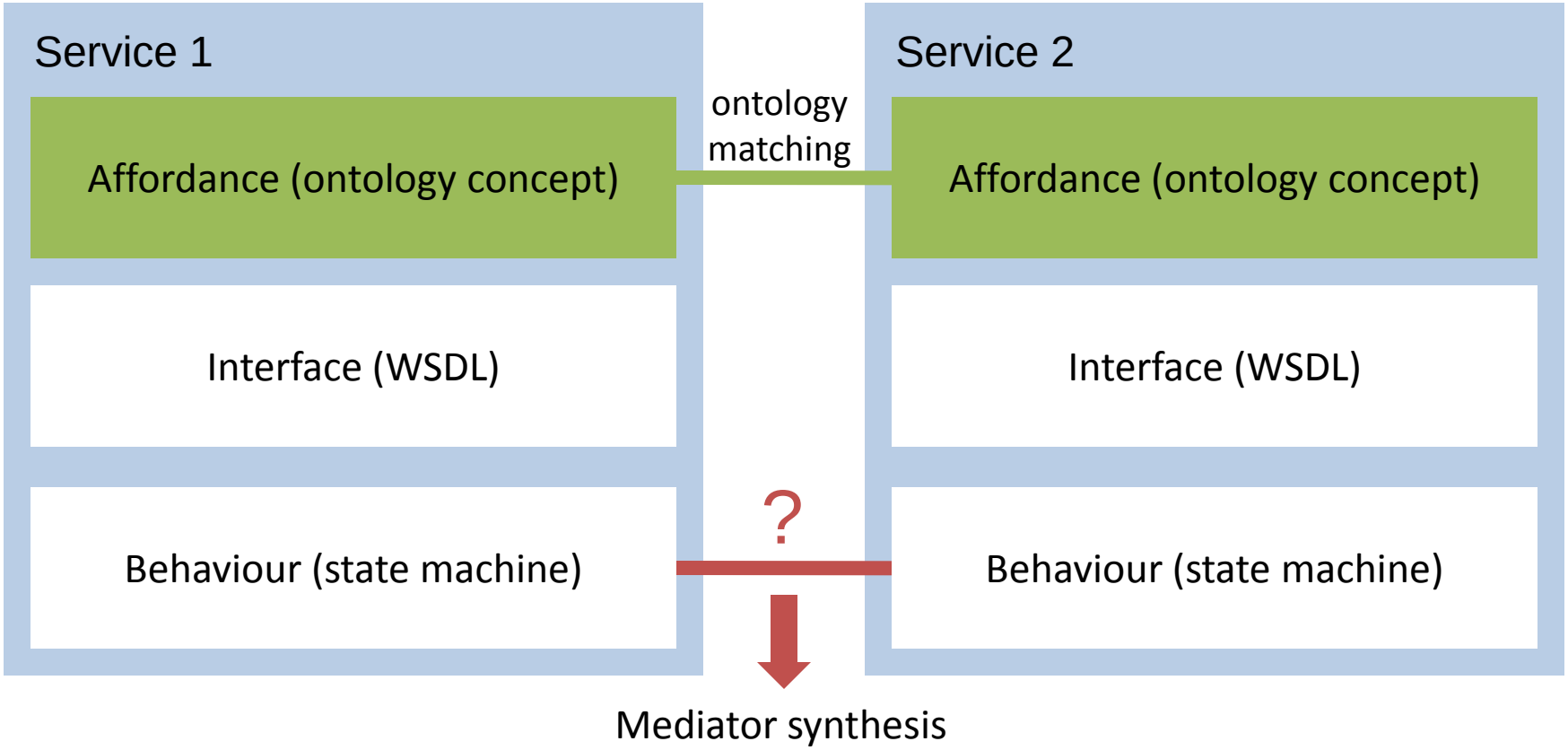
Goal model


User
Achieve[Arrange Travel]
Achieve[Arrange Entryl]
Achieve[Get AttractionTypes]
Achieve[Select Performance IF AttractionIsAPerformance]
Achieve[Get TravelTicket IF ModeNotFree]
Connect
Achieve[Buy Ticket IF AttractionNotFree]
Connect
<>executed(makeBooking)
User
User
User
User
Connect
Achieve[Buy Ticket IF AttractionNotFree]
Connect

Service model



Service selection



Mediator synthesis

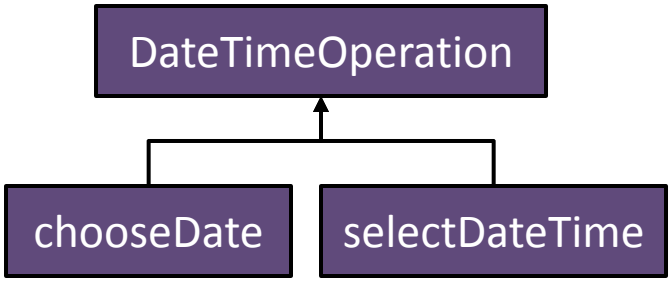
- Synthesise intermediary that enables communication between two differing protocols
- Such that the goal formula is achieved
  - $P_{S_1} \times M \times P_{S_2} \models G$
- Simple goal language: LTL operators plus
  - sent(c), received(c), executed(c)
  - Parameter and operation concepts in ontology
  - **<>executed(makeBooking)**

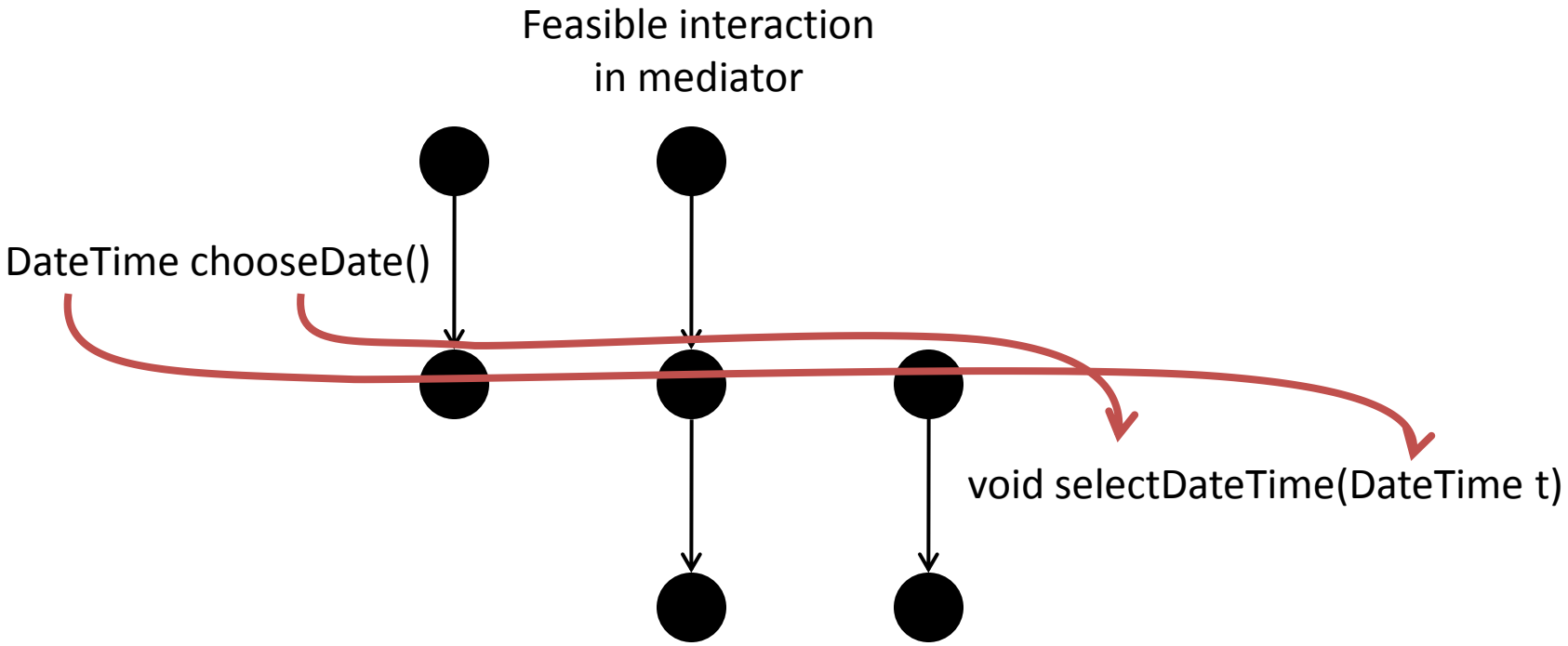
Mediator synthesis

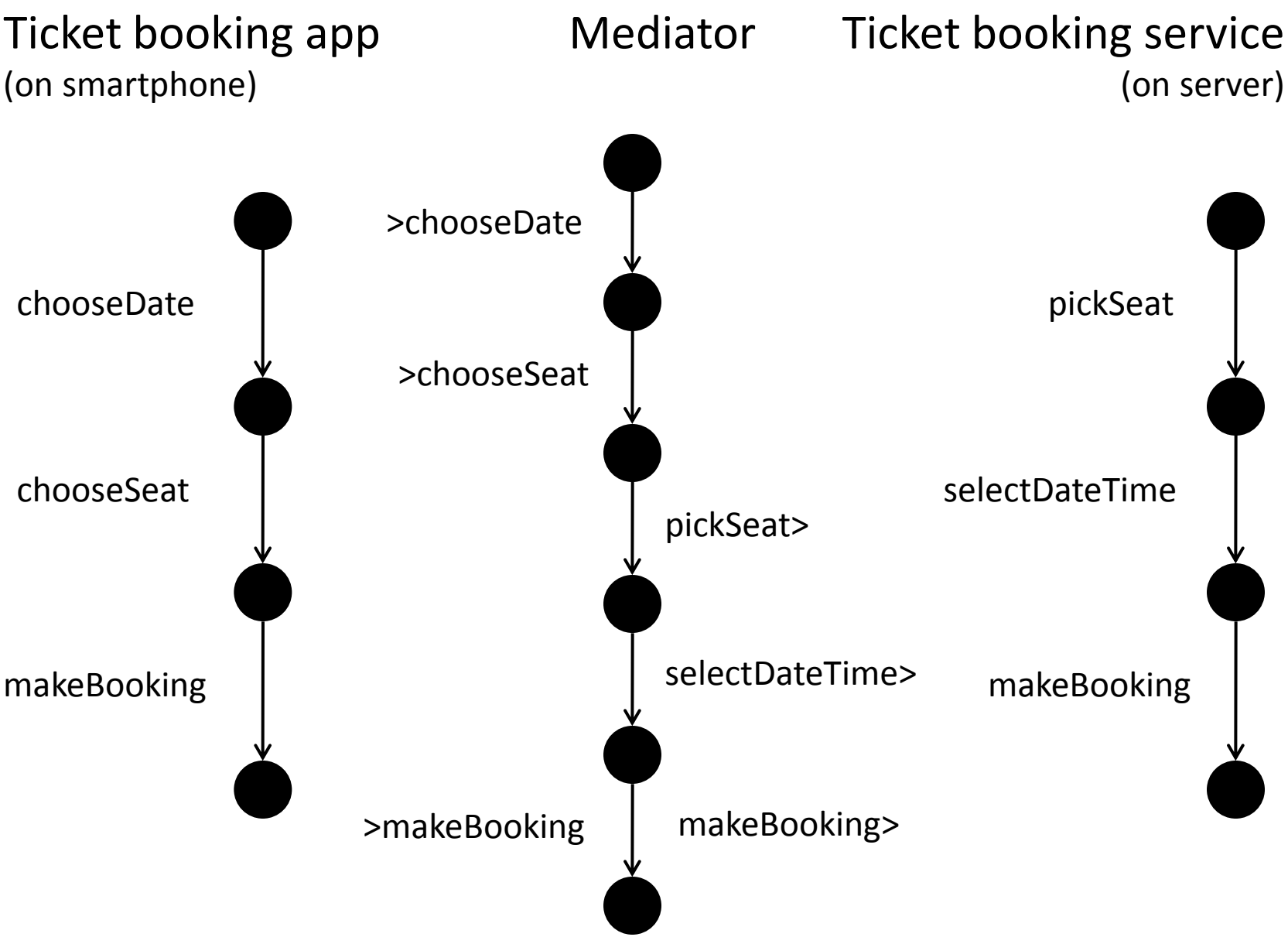
- Goal LTL compiled to Büchi automaton, reachability checked on parallel composition
- Path to goal must be a *feasible interaction*:
  - All input parameters are sent from one partner before being needed by the other partner
  - All output parameters are eventually provided
  - Permits operation re-ordering

- Services effectively synchronise on operations with matching ontology concepts

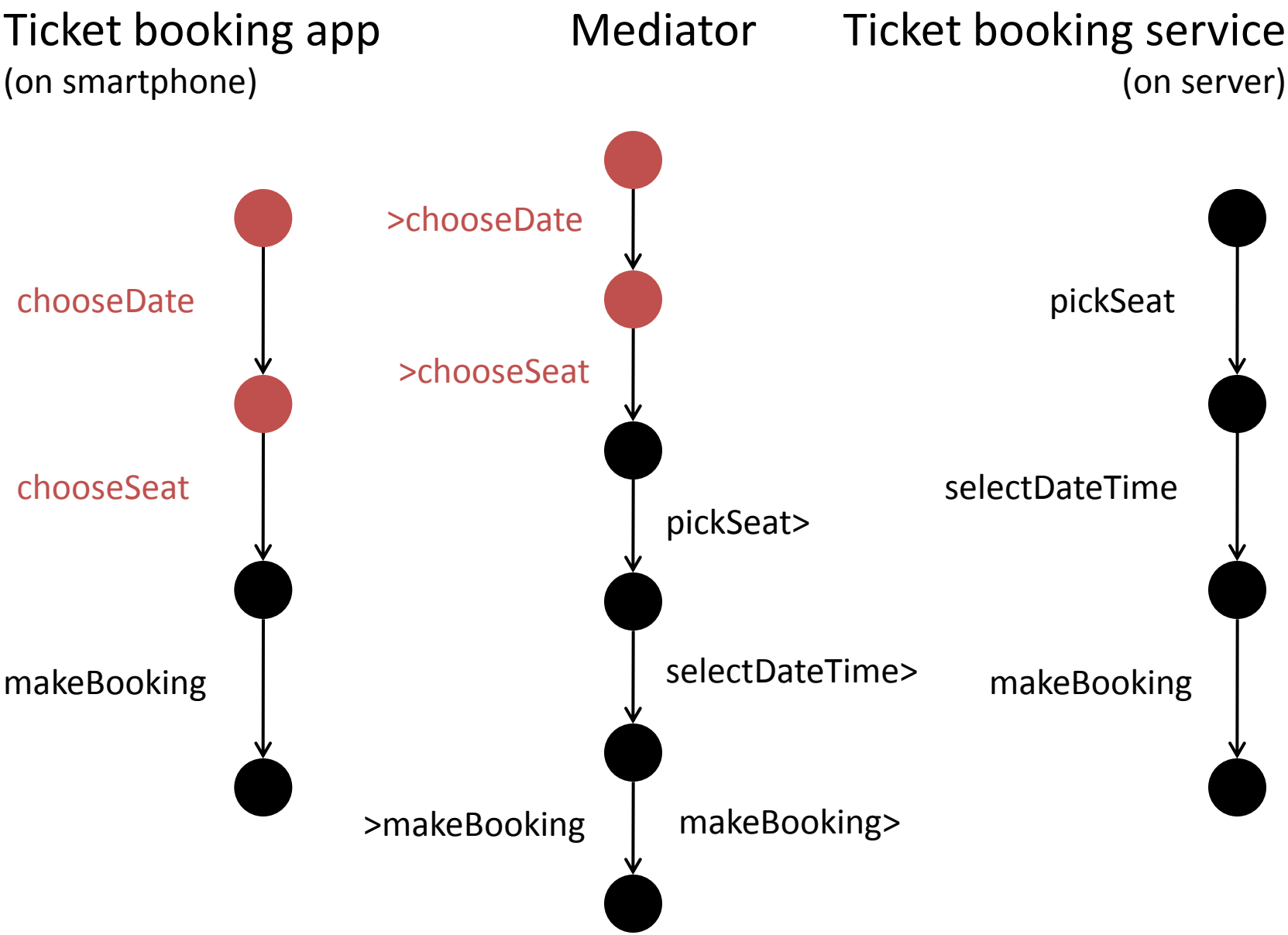
Ontology:



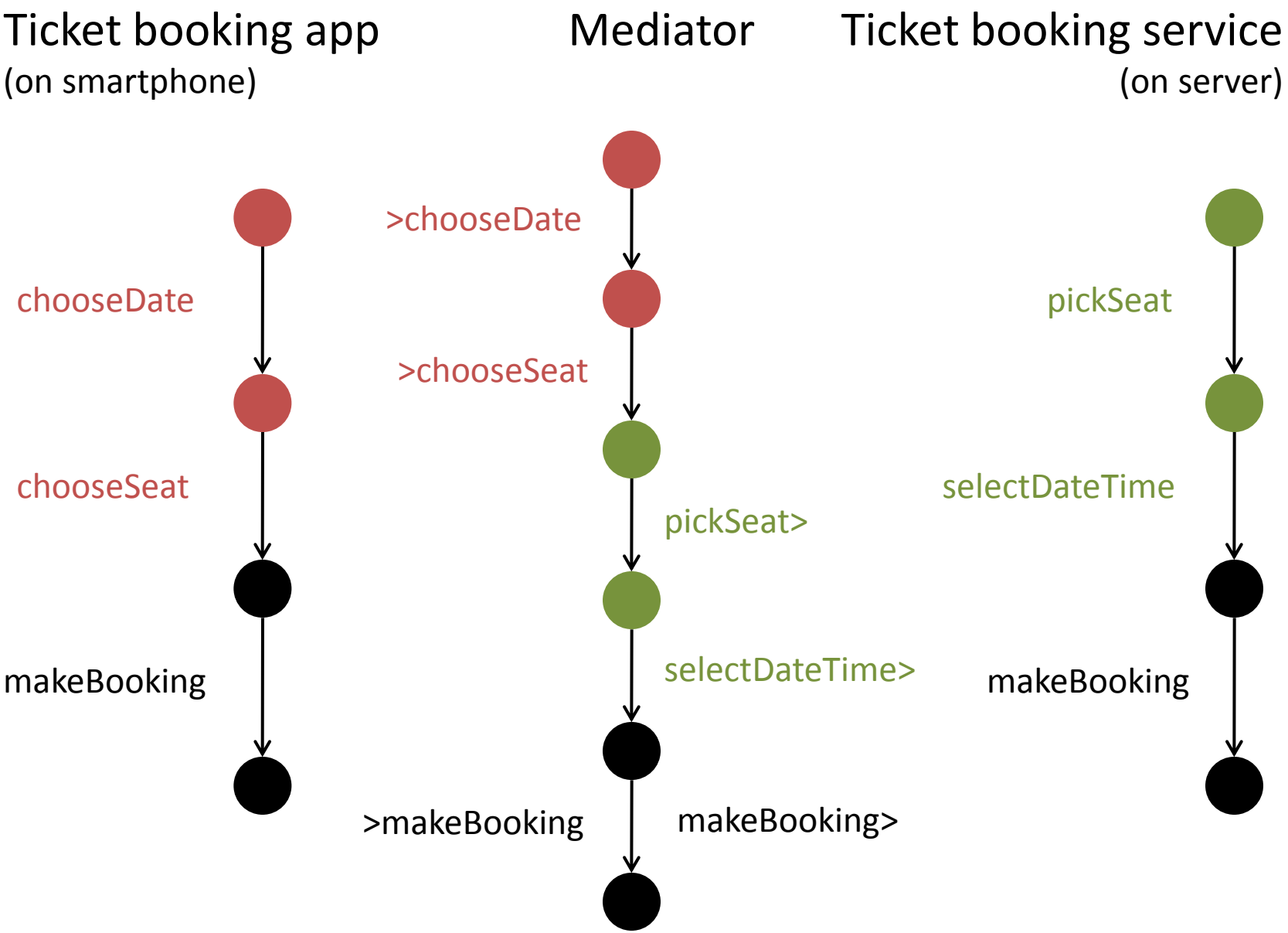




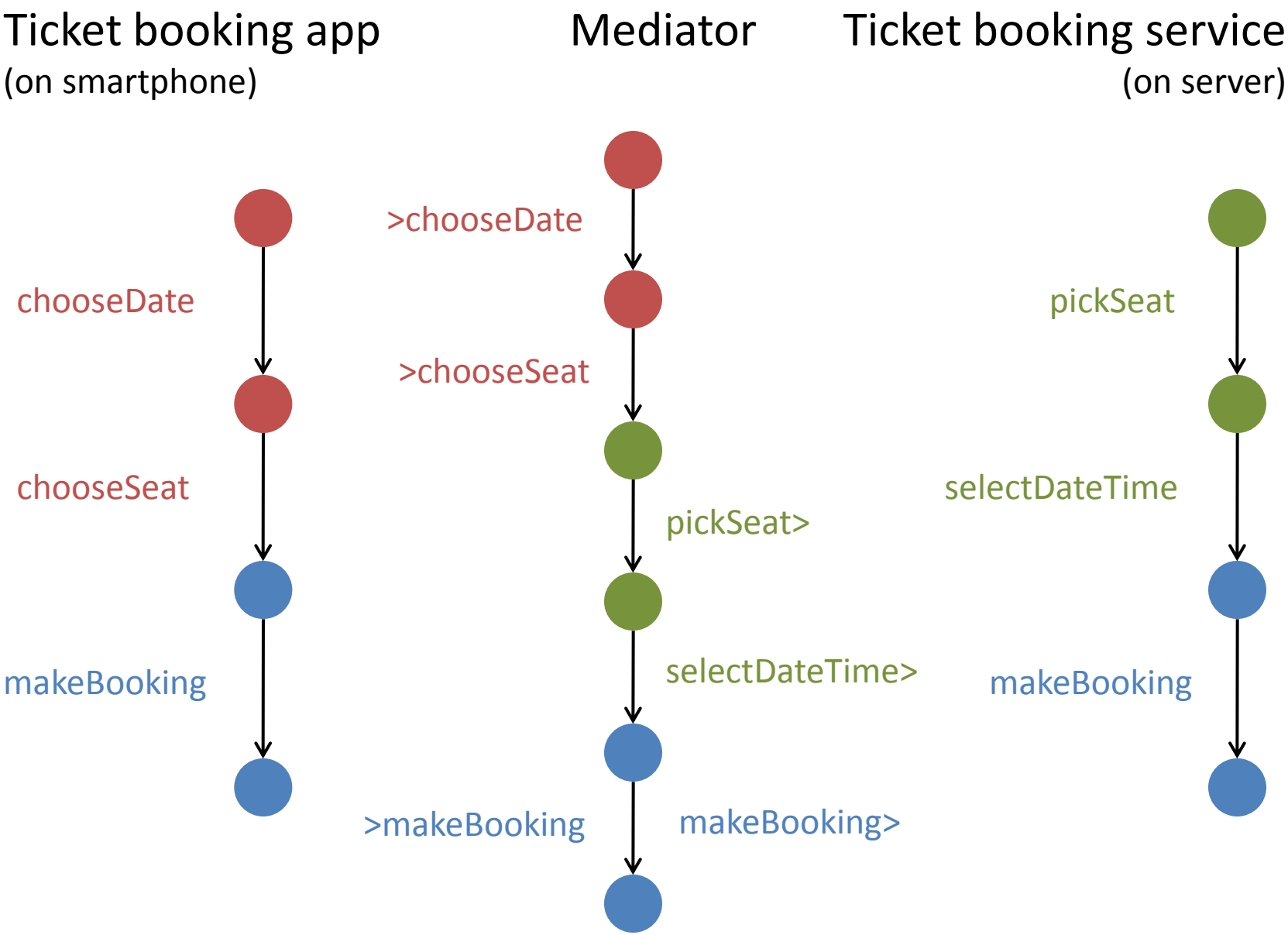
<>executed(makeBooking)



<>executed(makeBooking)



<>executed(makeBooking)



<>executed(makeBooking)

Summary

- Build compositions of multiple services discovered at runtime
- Services describe their interface and behaviour (runtime models)
- Synthesis overcomes signature and protocol mismatch
- Achieve goals specified using KAOS
- Future:
  - Consider non-functional properties
  - Relaxed goals guided by what can be realised given discovered services