

Verteilte Systeme

3. Dienstevermittlung

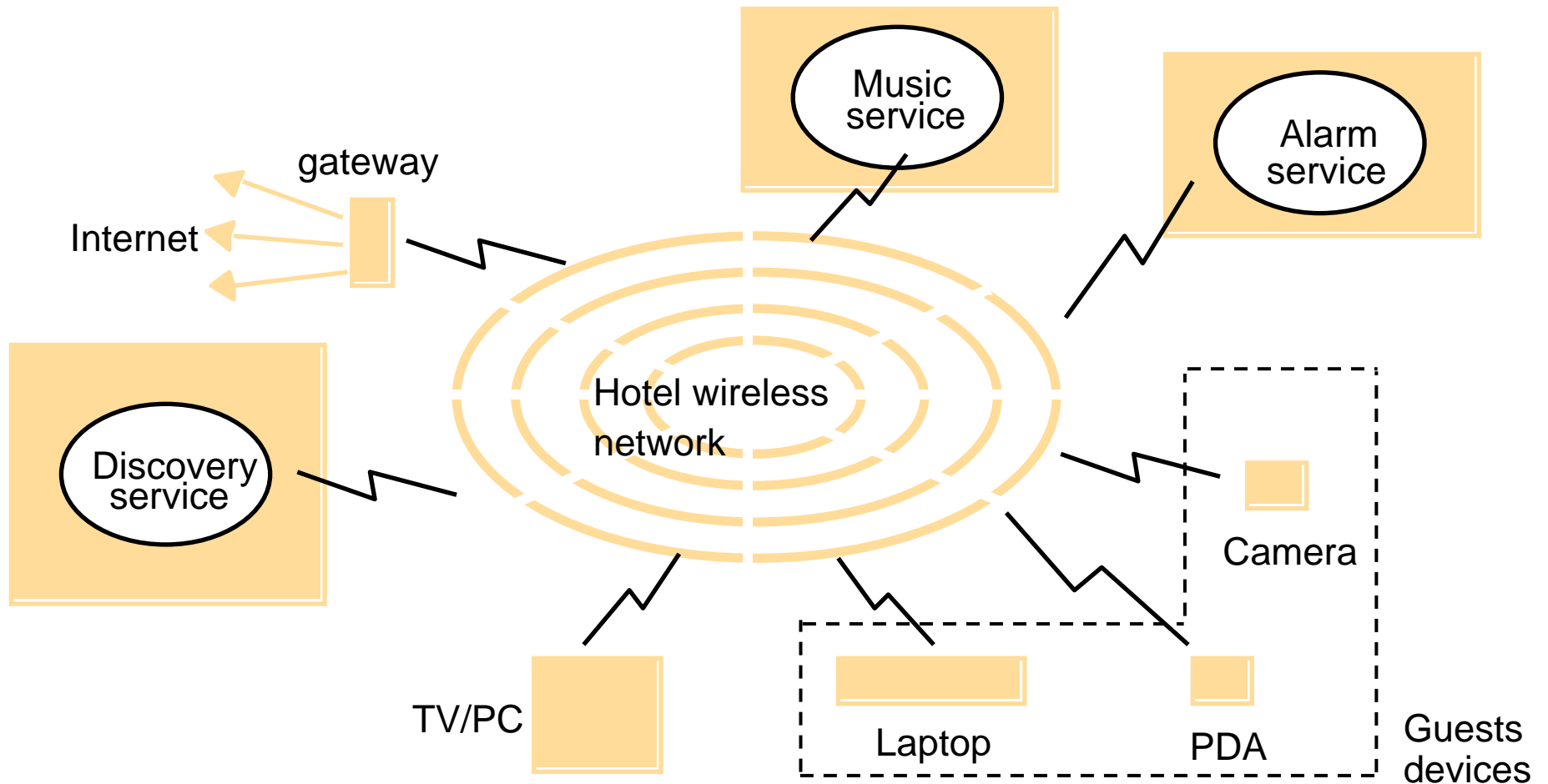
3.1 Aufgabenstellungen und Lösungstechniken

Sebastian Iwanowski
FH Wedel

Was versteht man unter Dienstevermittlung in verteilten Systemen ?

- Zusammenbringen von Fragen und Antworten
- Ermöglichung des Nutzens von Kombinationen verschiedener Dienste
- Ermöglichung spontanen Eintretens und Austretens von der Dienstplattform unter folgenden Regeln:
 - Client darf jederzeit eintreten oder austreten.
 - Server muss Dienst für bestimmte Zeit zusagen.
 - Server darf nur austreten,
wenn er gerade keinen Dienst für einen Client erbringt
und wenn die zugesagte Zeit abgelaufen ist.

Mögliches Szenario: Gast im Hotel



aus Coulouris et al.: *Verteilte Systeme*

Touristeninformationssystem als Beispiel für eine Dienstplattform mit weiter gehenden Ansprüchen

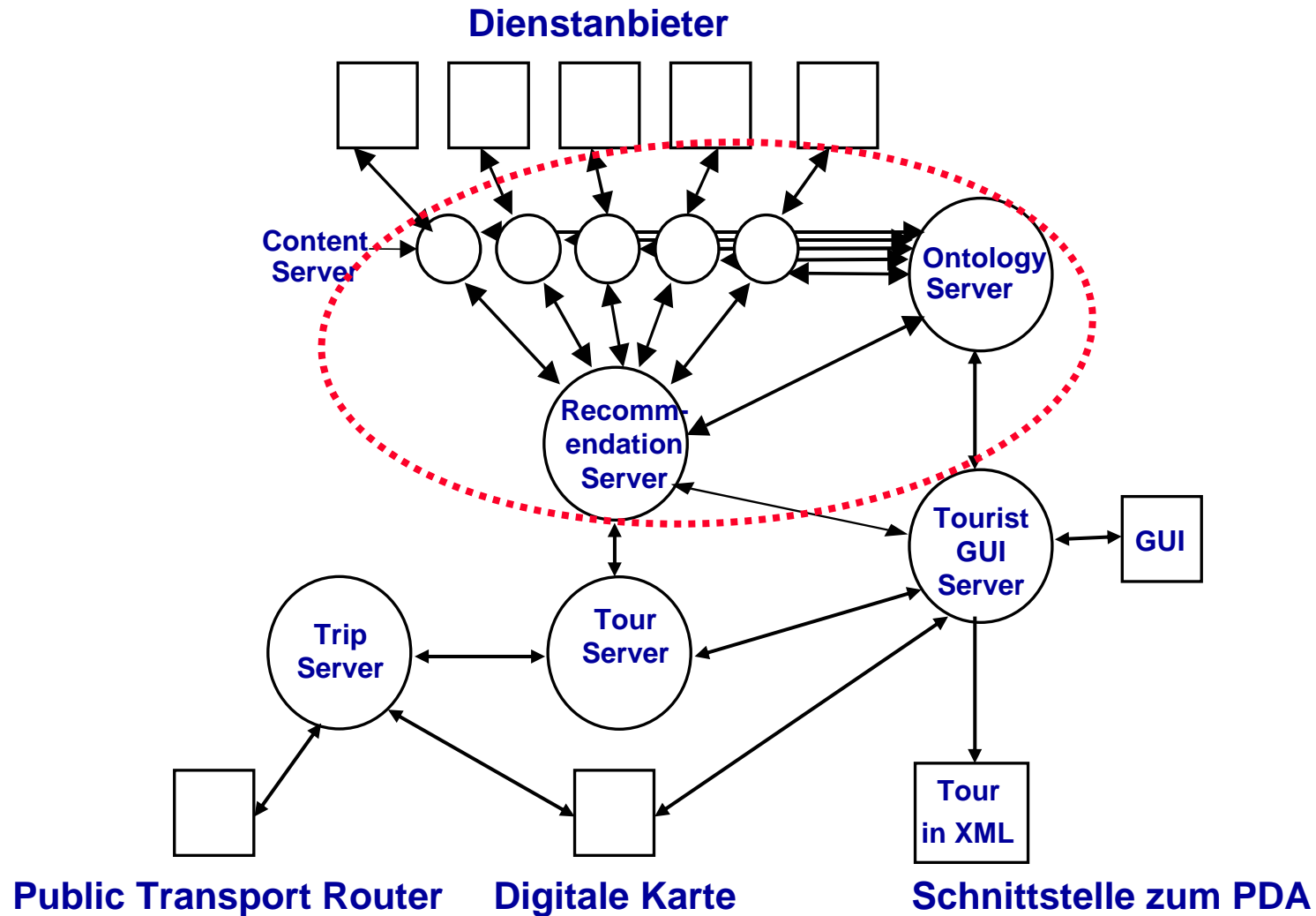
Ziele:

- Unterstützung plattform- und **software**unabhängiger Dienste
→ wird auch von WebServices angestrebt
- Unterstützung der Generierung neuer Dienstypen
- Unterstützung der Generierung von Mehrwertdiensten

Lösung (2000-2002):

- Verwendung von Ideen aus der agentenorientierten Programmierung
- **keine** Verwendung agentenorientierter Programmierwerkzeuge

Touristeninformationssystem



Ontologien

Was ist eine Ontologie ?

- Begriff „Ontologie“ stammt aus agentenorientierter Programmierung
- Ontologie bezeichnet Begriffswelt
- Ontologien dienen zur Beschreibung von Semantik
- Neueste Entwicklung: Semantic Web

Syntax zur Beschreibung von Ontologien

- Objektorientierte Klassifizierungsbäume (UML)
- KQML (Knowledge Query Manipulation Language)
- XML

Ontologien

Was bezweckt man mit der Definition von Ontologien ?

- **Vereinheitlichung der Begriffswelt zwischen autonomen Dienst- oder Informationsanbietern**
- **Möglichkeit der Einbindung neuer Anbieter in ein bestehendes verteiltes Dienst- oder Informationssystem**

Ontologien: Beispiel einer Tripbeschreibung

KQML:

```
<Trip> ::= (Trip
    :startingPlace <GeoPlace>      :departure <TimeStamp>
    :destinationPlace <GeoPlace>   :arrival <TimeStamp>           :numberOfChanges <number>
    :rides (list {<Ride>}+))
```

```
<Ride> ::= <PublicRide> | <PrivateRide>
```

```
<PublicRide> ::= (PublicRide
    :lineName <string>
    :startingPlace <StopPlace>      :departure <TimeStamp>
    :destinationPlace <StopPlace>   :arrival <TimeStamp>
    :directionPlace <StopPlace>     :rideTime <number>)
```

```
<PrivateRide> ::= (PrivateRide
    :transportMedium <string>
    :startingPlace <Place>           :departure <TimeStamp>
    :destinationPlace <Place>       :arrival <TimeStamp>
    :rideLength <number>           :rideTime <number>)
```


Ontologien: Beispiel einer Tripbeschreibung

XML: (informell)

```
<Trip> ::= <trip language="<String>" orderNr="<Integer>">  
  <starting> <GeoPlace> <Date> <Time> </starting>  
  <ending> <GeoPlace> <Date> <Time> </ending>  
  <rides> {<Ride>}+ </rides>  
</trip>
```

```
<Ride> ::= <PublicRide> | <PrivateRide>
```

```
<PublicRide> ::= <public-ride orderNr="<Integer>">  
  <line> <String> </line>  
  <starting>  
    <station-name> <String> </station-name>  
    <GeoPlace>  
    <Date>  
    <Time>  
  </starting>  
  <ending>  
    <station-name> <String> </station-name>  
    <GeoPlace>  
    <Date>  
    <Time>  
  </ending>  
</public-ride>
```

```
<PrivateRide> ::= <private-ride orderNr="<Integer>">  
  <starting> <GeoPlace> <Date> <Time> </starting>  
  <ending> <GeoPlace> <Date> <Time> </ending>  
</private-ride>
```

Ontologien: Beispiel einer Tripbeschreibung

XML: (WSDL) Teil 1

```
<complexType abstract="true" name="Ride">
  <sequence/>
</complexType>
<complexType name="Rides">
  <complexContent>
    <restriction base="soapenc:Array">
      <attribute ref="soapenc:arrayType" wsdl:arrayType="tns1:Ride[]"/>
    </restriction>
  </complexContent>
</complexType>
<complexType name="GeoPlace">
  <sequence>
    <element name="gkX" nillable="true" type="xsd:int"/>
    <element name="gkY" nillable="true" type="xsd:int"/>
    <element name="longitude" nillable="true" type="xsd:float"/>
    <element name="latitude" nillable="true" type="xsd:float"/>
  </sequence>
</complexType>
<complexType name="Date">
  <sequence>
    <element name="day" type="xsd:int"/>
    <element name="month" type="xsd:int"/>
    <element name="year" type="xsd:int"/>
  </sequence>
</complexType>
<complexType name="Time">
  <sequence>
    <element name="hour" type="xsd:int"/>
    <element name="minute" type="xsd:int"/>
  </sequence>
</complexType>
```

Ontologien: Beispiel einer Tripbeschreibung

XML: (WSDL) Teil 2

```
<complexType name="Situation">
  <sequence>
    <element name="geoPlace" nillable="true" type="tns1:GeoPlace"/>
    <element name="locationName" nillable="true" type="xsd:string"/>
    <element name="date" nillable="true" type="tns1:Date"/>
    <element name="time" nillable="true" type="tns1:Time"/>
  </sequence>
</complexType>
<complexType name="PrivateRide">
  <complexContent>
    <extension base="tns1:Ride">
      <sequence>
        <element name="endingSituation" nillable="true" type="tns1:Situation"/>
        <element name="startingSituation" nillable="true" type="tns1:Situation"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="PublicRide">
  <complexContent>
    <extension base="tns1:PrivateRide">
      <sequence>
        <element name="lineName" nillable="true" type="xsd:string"/>
        <element name="directionPlace" nillable="true" type="xsd:string"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

Ontologien: Beispiel einer Tripbeschreibung

XML: (WSDL) Teil 3

```
<complexType name="PublicRides">
  <complexContent>
    <restriction base="soapenc:Array">
      <attribute ref="soapenc:arrayType" wsdl:arrayType="tns1:PublicRide[]"/>
    </restriction>
  </complexContent>
</complexType>
<complexType name="EmptyRide">
  <complexContent>
    <extension base="tns1:Ride">
      <sequence/>
    </extension>
  </complexContent>
</complexType>
<complexType name="EmptyRides">
  <complexContent>
    <restriction base="soapenc:Array">
      <attribute ref="soapenc:arrayType" wsdl:arrayType="tns1:EmptyRide[]"/>
    </restriction>
  </complexContent>
</complexType>
<complexType name="PrivateRides">
  <complexContent>
    <restriction base="soapenc:Array">
      <attribute ref="soapenc:arrayType" wsdl:arrayType="tns1:PrivateRide[]"/>
    </restriction>
  </complexContent>
</complexType>
```

Ontologien: Beispiel einer Tripbeschreibung

XML: (WSDL) Teil 4

```
<complexType name="Trip">  
  <sequence>  
    <element name="rides" nillable="true" type="tns1:Rides"/>  
    <element name="startingSituation" nillable="true" type="tns1:Situation"/>  
    <element name="endingSituation" nillable="true" type="tns1:Situation"/>  
  </sequence>  
  <attribute name="language" type="xsd:string"/>  
</complexType>
```

Softwareunabhängige Dienstevermittlung

Problem:

- **Woher weiß ein Client, welche Fragen der Server versteht ?**
- **Woher weiß ein Server, welche Antwort ein Client zu einer gegebenen Frage erwartet ?**

Lösung:

- 1. Server gibt seine Ontologie bekannt**
- 2. Client stellt seine Frage als unvollständigen Begriff aus der Server-Ontologie**
- 3. Server vervollständigt den Begriff mit Daten, die er generiert, und sendet sie als Antwort zurück.**
 - Client kann in Frage angeben, wie viele Antworten er erwartet und wie detailliert die Antworten sein sollen.

Softwareunabhängige Dienstevermittlung

Beispiel für TripServer als Server mit der eben definierten Ontologie (KQML):

Frage des TourServers:

(Trip :startingPlace (GeoPlace :x 4591300 :y 5822100)
:departure (TimeStamp :date "29.02.2000" :time "12:00")
:destinationPlace (GeoPlace :x 4593100 :y 5825850))

Antwort des TripServers:

(Trip :startingPlace (GeoPlace :x 4591300 :y 5822100)
:departure (TimeStamp :date "29.02.2000" :time "12:05")
:destinationPlace (GeoPlace :x 4593100 :y 5825850)
:arrival (TimeStamp :date "29.02.2000" :time "12:17"))

Detailliertere Frage des TourServers:

(Trip :startingPlace (GeoPlace :x 4591300 :y 5822100)
:departure (TimeStamp :date "29.02.2000" :time "12:00")
:destinationPlace (GeoPlace :x 4593100 :y 5825850)
:rides *)

Detailliertere Antwort des TripServers:

(Trip :startingPlace (GeoPlace :x 4591300 :y 5822100)
:departure (TimeStamp :date "29.02.2000" :time "12:05")
:destinationPlace (GeoPlace :x 4593100 :y 5825850)
:arrival (TimeStamp :date "29.02.2000" :time "12:17")
:numberOfChanges 2
:rides
 ((PrivateRide :transportMedium "footwalk"
 :startingPlace (GeoPlace :x 4591300 :y 5822100) :departure (TimeStamp :date "29.02.2000" :time "12:05")
 :destinationPlace (GeoPlace :x 4591207 :y 5822200) :arrival (TimeStamp :date "29.02.2000" :time "12:07")
 :rideLength 137 :rideTime "00:02")
 (PublicRide :lineName "U9"
 :startingPlace (StopPlace :stopArea "U Turmstraße") :departure (TimeStamp :date "29.02.2000" :time "12:08")
 :destinationPlace (StopPlace :stopArea "U Osloer Straße") :arrival (TimeStamp :date "29.02.2000" :time "12:16")
 :directionPlace (StopPlace :stopArea "U Osloer Straße") :rideTime "00:08")
 (PrivateRide :transportMedium "footwalk"
 :startingPlace (GeoPlace :x 4593092 y: 5825754) :departure (TimeStamp :date "29.02.2000" :time "12:16")
 :destinationPlace (GeoPlace :x 4593100 :y 5825850) :arrival (TimeStamp :date "29.02.2000" :time "12:17")
 :rideLength 96 :rideTime "00:01"))

Einklinken eines neuen Anbieters in ein bestehendes verteiltes Informationssystem

Problem:

- **Wie kann sich ein neuer Anbieter so einbinden, dass er von potentiellen Clients auch gefunden wird ?**
- **Wie kann der Anbieter erreichen, dass ein potentieller Client seine Dienste versteht ?**

Lösung:

- 1. Neuer Anbieter meldet sich bei zentraler Stelle an, die von vielen Clients besucht wird.**
- 2. Neuer Anbieter benutzt eine im bestehenden Informationssystem bereits verwendete Ontologie**
- 3. Neuer Anbieter hängt von ihm benötigte neue Begriffe als Unterbegriffe an bestehende Begriffe an.**

Einklinken eines neuen Anbieters in ein bestehendes verteiltes Informationssystem

Beispiel:

- **Eine Menge von Ontologien für POIs ist dem System bereits bekannt**
- **Ein neuer ShoppingServer für Einkaufsmöglichkeiten will sich einbinden**
- **Ontologie des ShoppingServers besteht aus einem neuen Begriff ShoppingPOI mit seinen Unterbegriffen und Attributen**
- **ShoppingServer beschließt, seinen Begriff als Unterbegriff des bereits bestehenden Begriffs POI einzuklinken**

Einklinken eines neuen Anbieters in ein bestehendes verteiltes Informationssystem

Beispiel:

Bereits bekannte Ontologie:

(POI (DictEntry POI)

:subClasses (<SightseeingPOI> <DiningPOI> <EntertainmentPOI>)

:name (DictEntry Name) <string>

:address (DictEntry Address) <AddressPlace>

:description (DictEntry Description) <DictEntry>

:place (DictEntry Dummy) ({<GeoPlace>}+))

Einklinken eines neuen Anbieters in ein bestehendes verteiltes Informationssystem

Beispiel:

Ontologie des neuen ShoppingServers:

```
(ShoppingPlace (DictEntry ShoppingPlace)
  :subClasses
    ((DepartmentStore (DictEntry DepartmentStore) :subClasses ())
     (GroceryStore (DictEntry GroceryStore) :subClasses ())
     (FashionShop (DictEntry FashionShop) :subClasses ())
     (SouvenirShop (DictEntry SouvenirShop) :subClasses ())
     (SpecialShop (DictEntry SpecialShop) :subClasses ()))
  :sellingItems (DictEntry SellingItems) ({{<DictEntry>}}))
```

Anbieten eines Mehrwertdienstes

Problem:

- Einzelne Anbieter können verschiedene Dienste anbieten, aber keiner kann die Dienste kombinieren
- Client möchte mehrere Dienste in Anspruch nehmen, fordert aber, dass die Dienste koordiniert werden

Lösung:

- Ein Mehrwertanbieter bietet sich als Koordinator an.
- Der Mehrwertanbieter bietet keine originären Dienste an, sondern vermittelt Anfragen an die Teilanbieter.
- Der Mehrwertanbieter kennt alle Ontologien der Teilanbieter sowie die Beziehungen zwischen ihnen.

Anbieten eines Mehrwertdienstes

Beispiel:

- Der **RecommendationServer** bietet die Möglichkeit, nach Restaurants eines bestimmten Typs zu fragen, in denen ein Konzert eines bestimmten Typs stattfindet
- Ein **RestaurantServer** bietet Ontologie an, in der Restaurants katalogisiert sind
- Ein **EntertainmentServer** bietet Ontologie an, in der Entertainments katalogisiert sind
- Der RecommendationServer fragt diese beiden Server getrennt und übermittelt die zusammengesetzte Antwort, wobei er die jeweilige Herkunft nicht verbirgt

Anbieten eines Mehrwertdienstes

Beispiel:

Ontologie des RestaurantServers (Auszug):

(DiningPlace (DictEntry DiningPlace)

:subClasses

((Bistro (DictEntry Bistro)

:subClasses ()

:seatingFacilities (DictEntry SeatingFacilitiesExist) <boolean>

(**Restaurant** (DictEntry Restaurant)

:subClasses ()

:seats (DictEntry NumberOfSeats) <number>

:stars (DictEntry CookingHats) <number>))

:**cuisine** (DictEntry Cuisine) <DictEntry>

Anbieten eines Mehrwertdienstes

Beispiel:

Ontologie des EntertainmentServers (Auszug):

```
(EntertainmentPOI (DictEntry EntertainmentPOI)  
  :subClasses ()  
  :events (DictEntry Dummy) ({<CulturalEvent>}+)
```

```
<CulturalEvent> ::=  
(CulturalEvent (DictEntry CulturalEvent)  
  :subClasses (<Theater> <Concert>)  
  :description (DictEntry Description) <dictEntry>  
  :admissionFee (DictEntry AdmissionFee) <float>  
  :startingTime (DictEntry StartingTimeGeneral) <TimeStamp>  
  :endingTime (DictEntry EndingTimeGeneral) <TimeStamp>)))
```

```
<Concert> ::=  
(Concert (DictEntry Concert)  
  :subClasses  
    ((Classic (DictEntry Classic) :subClasses ())  
     (Jazz (DictEntry Jazz) :subClasses ())  
     (Folk (DictEntry Folk) :subClasses ())  
     (Pop (DictEntry Pop) :subClasses ())  
     (Rock (DictEntry Rock) :subClasses ())))))
```


Anbieten eines Mehrwertdienstes

Beispiel:

Frage an den RecommendationServer:

(POI ((Restaurant :cuisine Italian) AND
(EntertainmentPOI :events (Jazz :startingTime (TimeStamp :date „29.02.2000“))))))

Frage des RecommendationServers an den RestaurantServer:

(POI (Restaurant :cuisine Italian))

Frage des RecommendationServers an den EntertainmentServer:

(POI (EntertainmentPOI :events (Jazz :startingTime (TimeStamp :date „29.02.2000“)))

Anbieten eines Mehrwertdienstes

Beispiel:

Antwort des RecommendationServers nach Erhalt der Teilantworten:

```
((Restaurant :seats 50 :stars 3 :cuisine (DictEntry Italian) :name "Piazza del Jazz"  
:address (AddressPlace :postalCode 10400 :city "Berlin"  
:streetName "Käthe-Kollwitz-Platz" :number 3)  
:description (DictEntry PiazzaJazzEntry)  
:place ((GeoPlace :x 4000 :y 5500))))
```

AND

```
(EntertainmentPOI :seats 50 :name "Piazza del Jazz"  
:address (AddressPlace :postalCode 10400 :city "Berlin"  
:streetName "Käthe-Kollwitz-Platz" :number 3)  
:description (DictEntry PiazzaJazzEntry)  
:place ((GeoPlace :x 4000 :y 5500))  
:events ((Jazz  
:description (DictEntry AckermanConcertEntry)  
:startingTime (TimeStamp :date „29.02.2000“ :time „20:00“)  
:endingTime (TimeStamp :date „29.02.2000“ :time „23:00“))))))
```