ONTOLOGY-DRIVEN WEB SERVICES COMPOSITION TECHNIQUES

RUOYAN ZHANG

Advisor: Dr. I. Budak Arpinar
Committee: Dr. Hamid Arabina
Dr. Amit Sheth

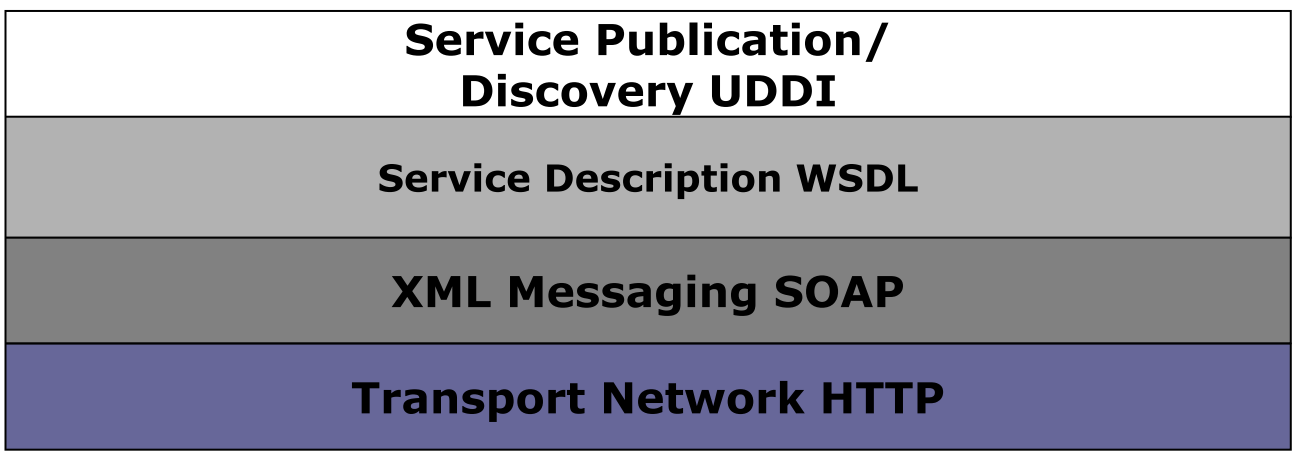
Presentation Layout

- Background on Web Services
- Challenges for Web Services Composition
- Interface-Matching Automatic (IMA) Composition
- Human-Assisted (HA) Composition
- Conclusions and Future Work

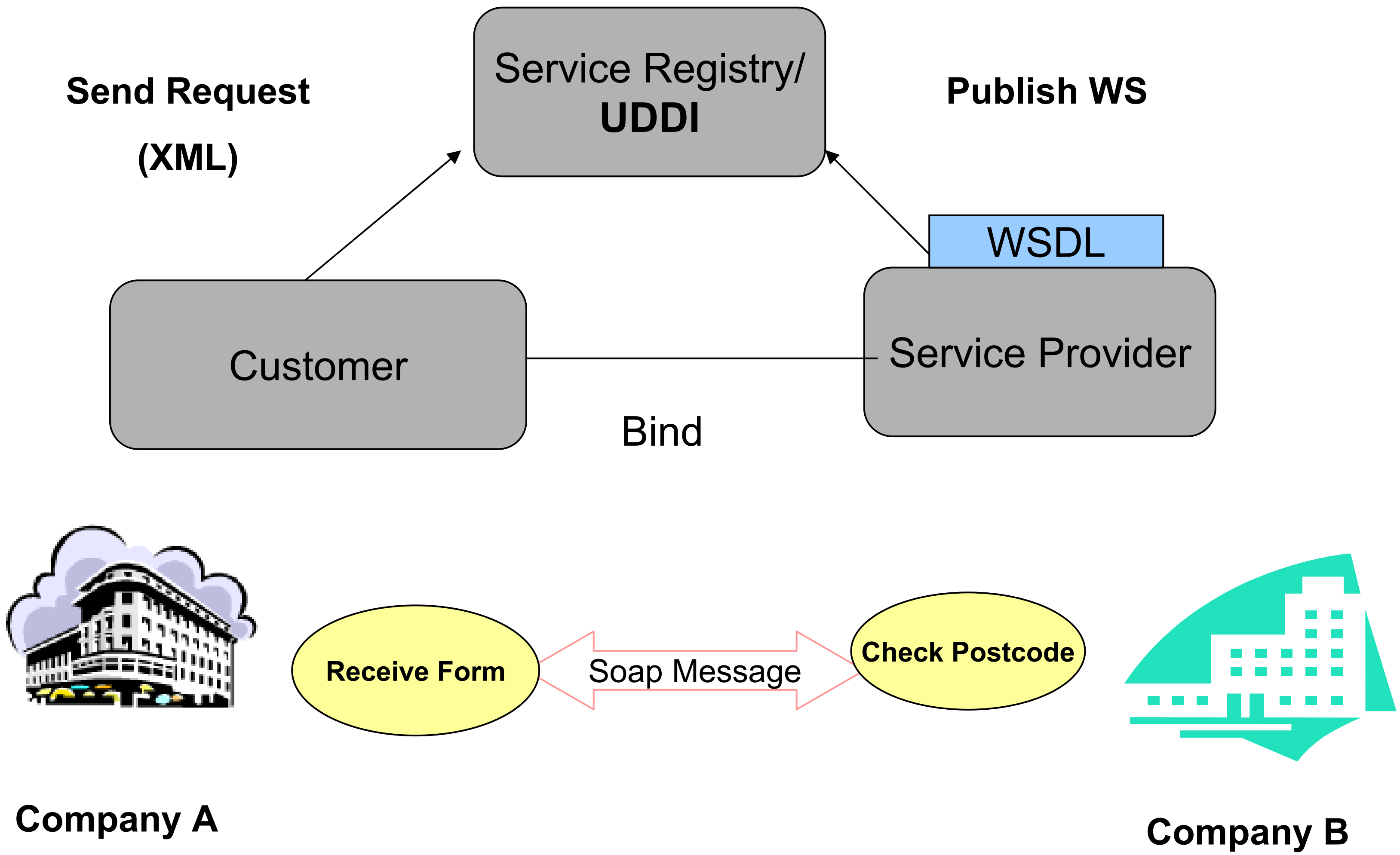
Web Service

- A Web Service is a software application identified by a URI, whose interfaces and binding are capable of being defined, described and discovered by XML artifacts and supports direct interactions with other software applications using XML based messages via Internet-based protocols (W3C definition).
- A self-contained, self-described, and self-advertised composition unit (application/ component).

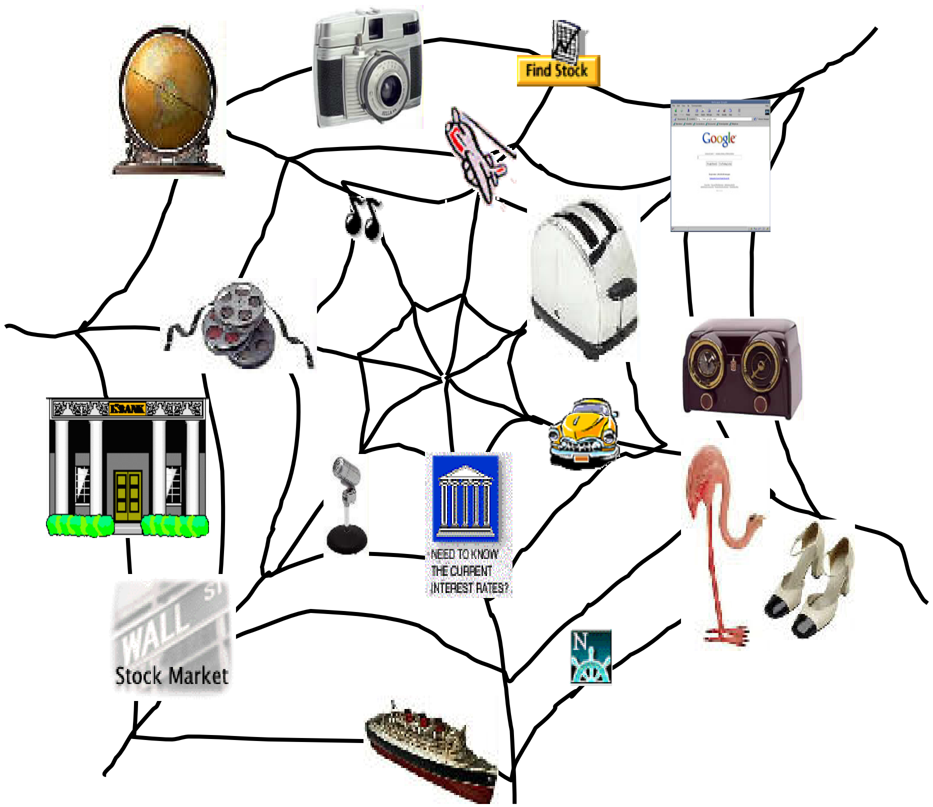
Web Services Stack



How they work and when we need them?

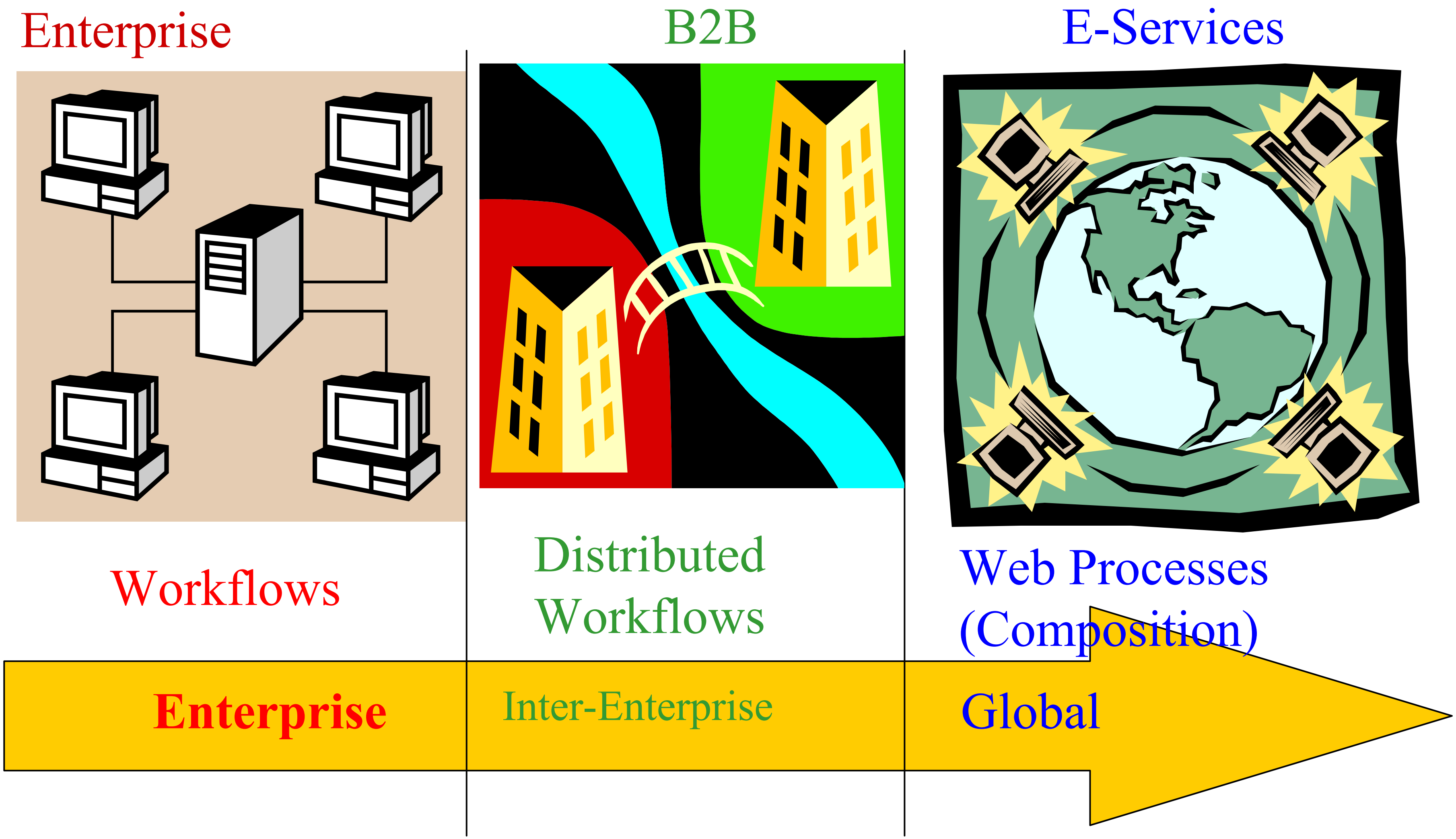


Web is turning into a collection of Web Services



- The number of companies that have completed an IT project involving Web services standards has **grown** in a survey released in 2003 [TechWeb].
- The percentage of surveyed firms using standards such as XML or SOAP increased from **11** percent in mid-2002 to **31** percent 2003, market researcher Forrester Research Inc.

Globalization of Web processes

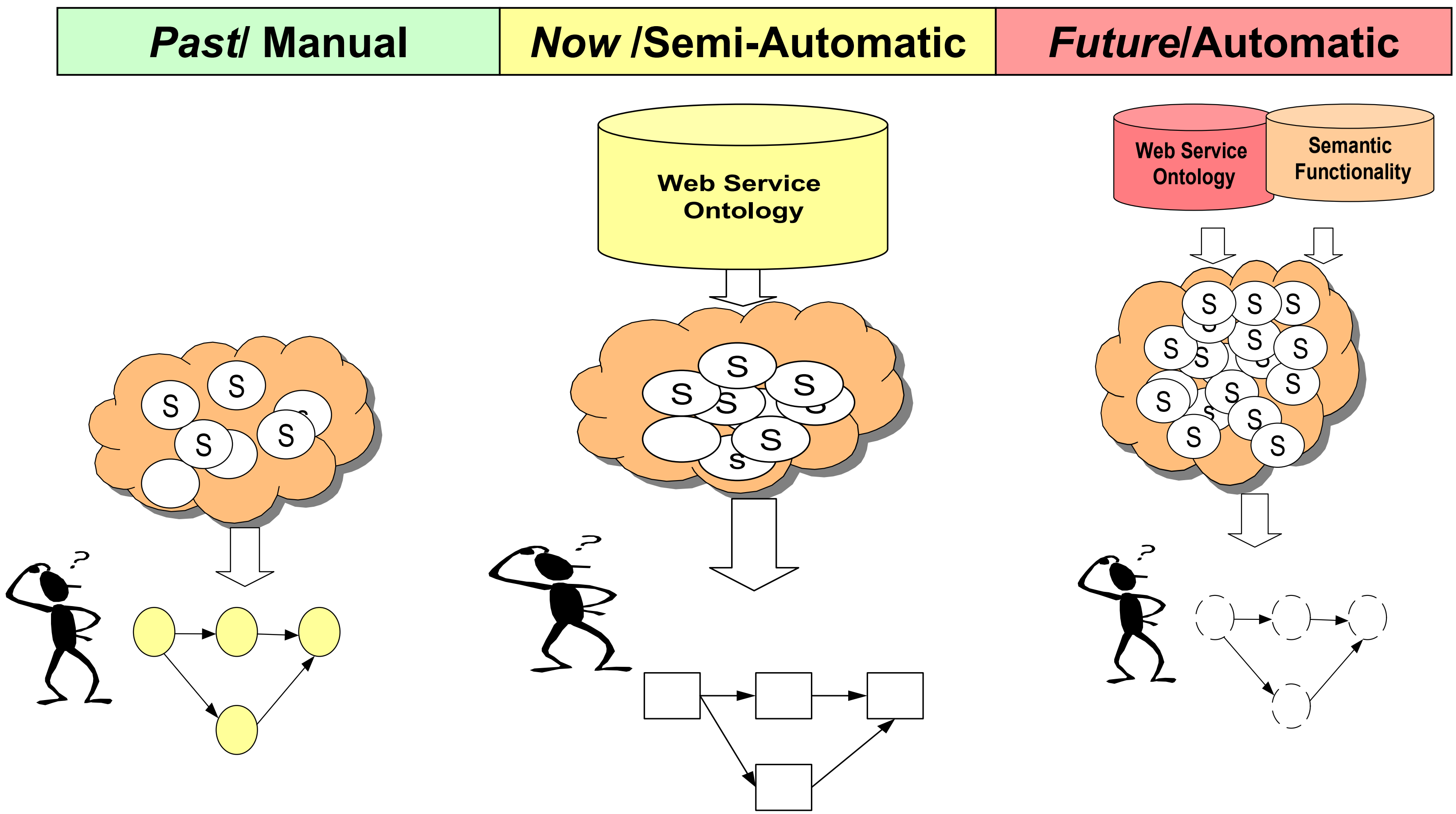


[Ref: Sheth, Cardoso WSTutorial]

Application integration/Web service composition

- An *Infoworld* survey shows that application integration costs are at least **25%** of the total IT budget at many companies.
- Gartner Dataquest predicts spending on integration projects will reach a staggering **$10.6** billion in 2006 .
- The same survey indicated that **55%** of the IT managers polled said Web services will make integration projects more viable.
- Why Web Service?
  - Open standards
  - Widespread support and universal access
  - Platform-neutral
  - (Hopefully) 0-line application development (i.e., automatically composed Web Process)

Composition ideas



Composition Challenges

- **Heterogeneity and Autonomy**
  - Syntactic, semantic and pragmatic
  - Complex rules/regulations related to B2B and e-commerce interactions
  - Solution: Machine processable descriptions
- **Dynamic** nature of business interactions
  - Demands: Efficient Discovery, Composition, etc.
- **Scalability** (Enterprises $\rightarrow$ Web)
  - Needs: Automated service discovery/selection and composition

Semantics is the most important enabler to address these challenges

[Ref: Sheth, Cardoso WS Tutorial]

More challenges

- Challenges of
  - capturing relations among services semantically (e.g., interface matching, complimentary function etc.),
  - modeling functionalities semantically,
  - developing efficient filtering mechanisms based on user preferences/context,
  - finding an optimal composition among alternatives through quality metrics.

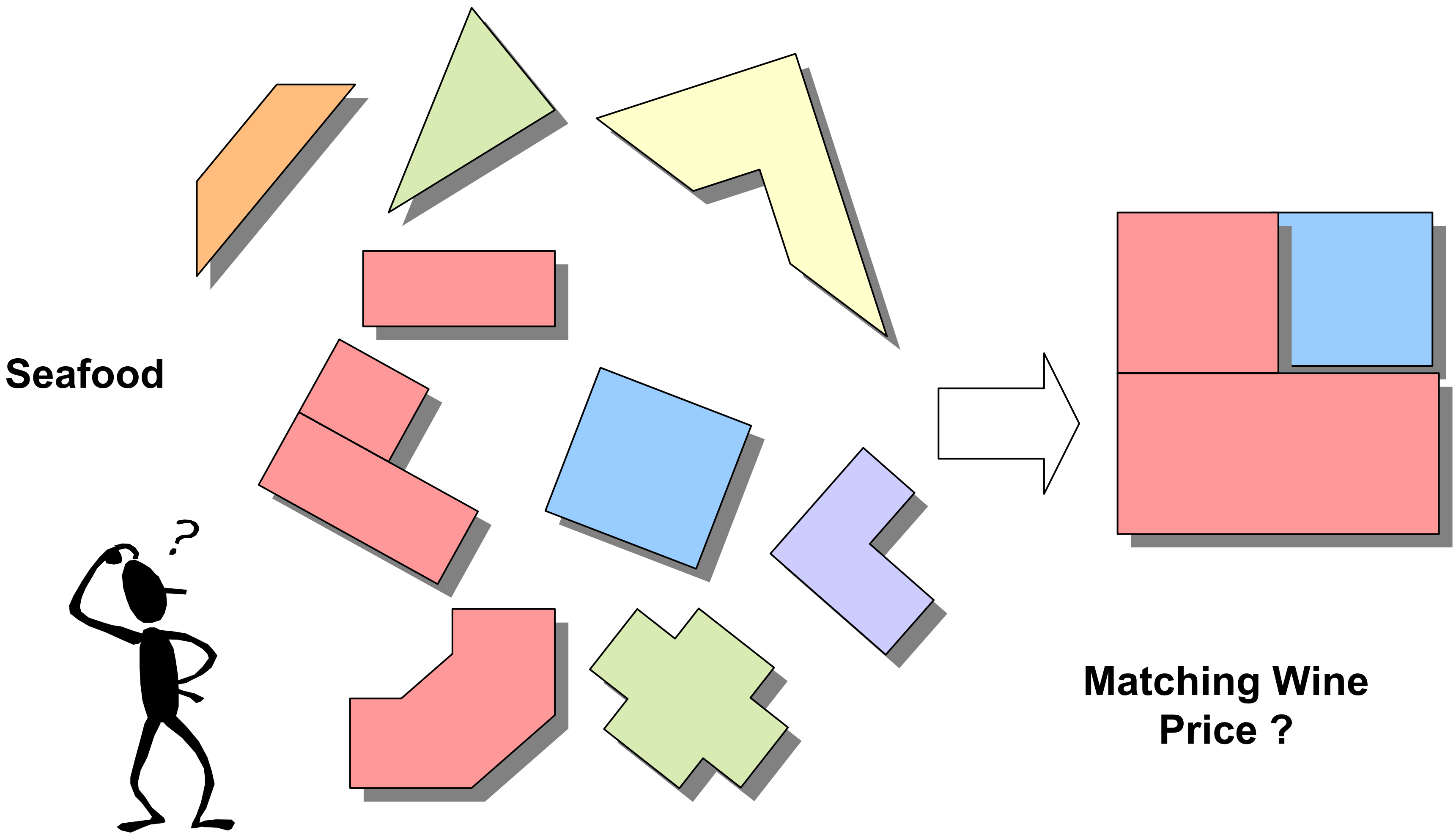
Web Service Composition: Industry

- WSDL + BPEL4WS
  - Interface description in WSDL
  - BPEL4WS specifies the roles of each of the partners and the logical flow of message (sequence, switch, while).
  - Error handling and message correlation.
  - They don’t tell much for generating a composition yet they can model a composition in terms its data and control flow.

Web Service Composition: Semantic-Web Community

- METEOR -S
  - Semantic annotation, discovery, composition of WSs
  - Data, Operational, Functional, and QoS Semantics
- Golog (AI Planning)
  - A method is presented to compose Web Services by applying logical inferencing techniques on pre-defined plan templates [Mcliraith & Son, 2002].
  - Semantic/ontological representation of states, actions, goals, and events are needed.
  - How to specify pre- and post-conditions in an explicit way by referring to structural properties of incoming and outgoing messages and internal state of the BPEL4WS process [Sritastava03].

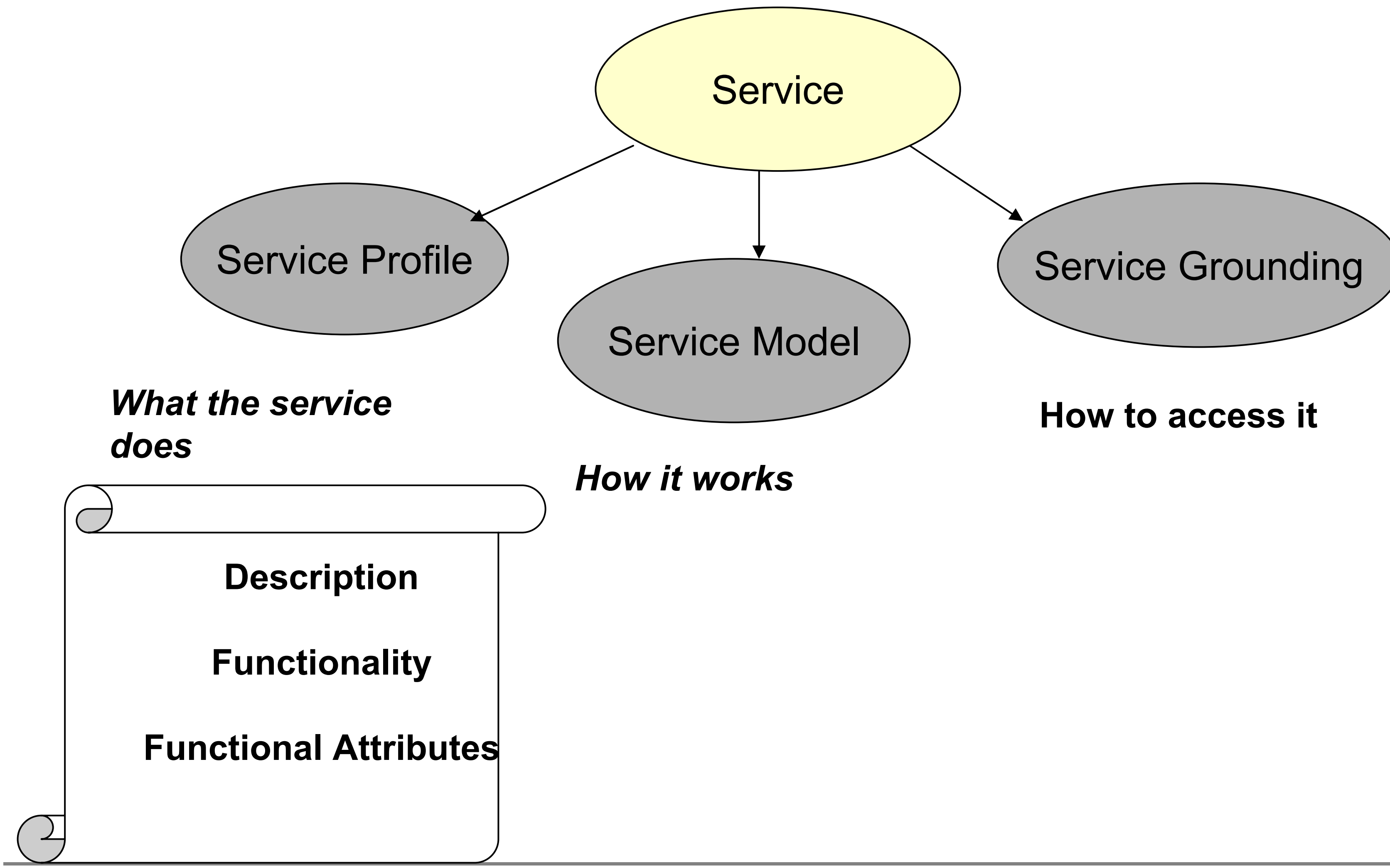
Automatic Composition



Automatic Composition Technique

- No predefined composition template
- Web services are assembled through a forward - chaining method
  - Interface relations (i.e., matching) with different weights are computed among WS interfaces as well as user I/Os and WS interfaces.
  - Ontological measures are used for matching
  - A WS net is generated for finding an optimal path among various compositions
- Adapted Bellman-Ford Shortest path (dynamic programming) algorithm. (Multiple inputs and outputs)
- Exploited DAML-S WS descriptions

Web Service Modeling: DAML-S



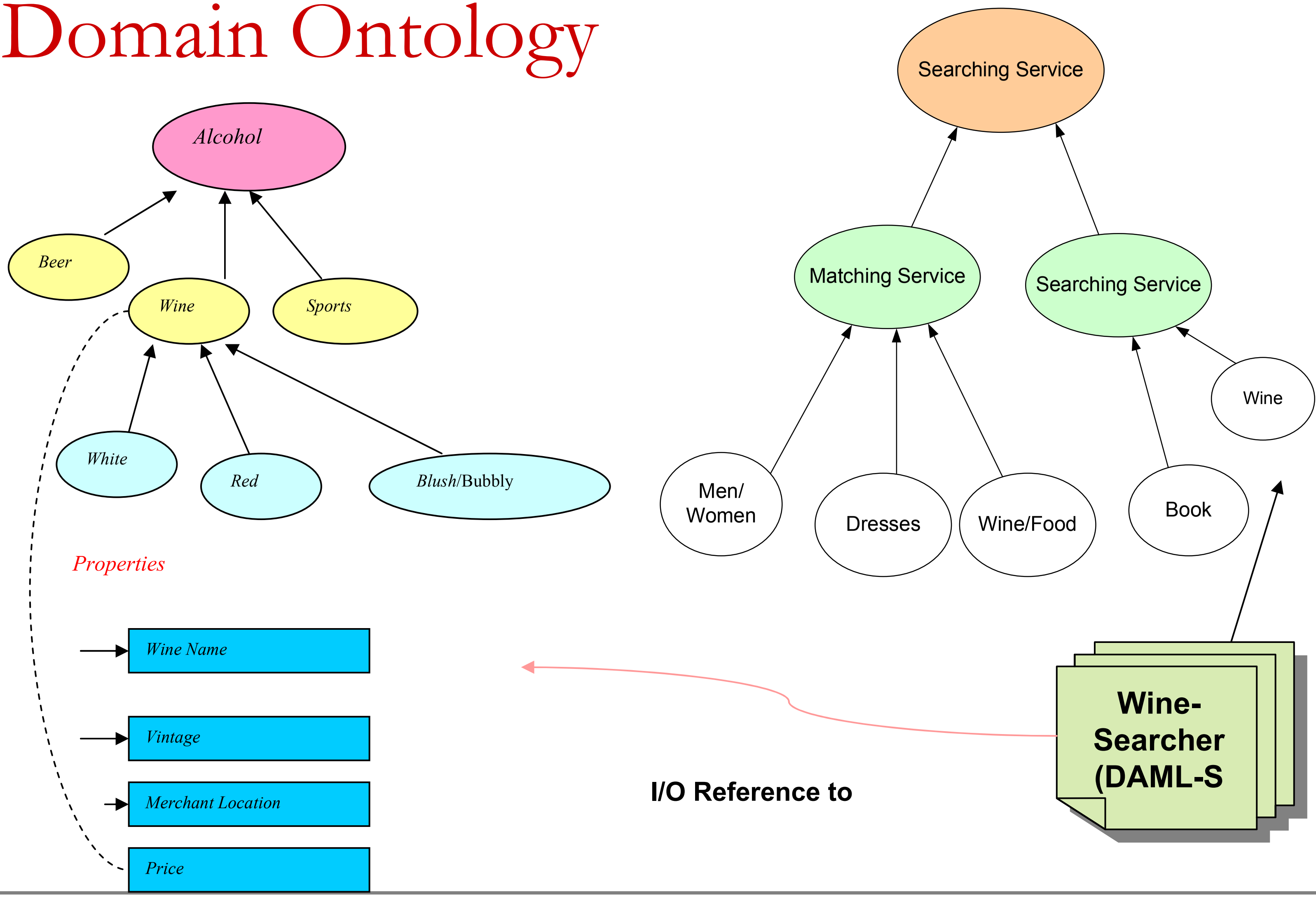
Wine-Search Service in DAML-S

```
<profileHierarchy: Wine-Search rdf: ID="Profile-Wine-Searcher">

<service: presentedBy rdf: resource="wine- searcher.owl#wine-searcher" />
<profile: has_process rdf: resource="wine-searcher- Process. owl# Wine-Searcher
    ProcessModel"/>

<profile: serviceName> Wine-Searcher.com</profile: serviceName>
<profile:textDescription>Wine-Searcher helps ….. database </profile:textDescription>
```

```
<profile: qualityRating>
<profile: qualityRating rdf:ID="wine-search-Rating">
<profile: ratingName>very good</profile: ratingName>
<profile: rating rdf:resource="owl-s/1.0/Concepts.owl#GoodRating">
 </profile: QualityRating>
```

```
<profile: hasInput  rdf:resource="Service-Concept.owl#wineName" />
<profile: hasOutput rdf:resource="Service-Concept. owl# winePrice" />

</profileHierarchy: Wine-Search Service>
```

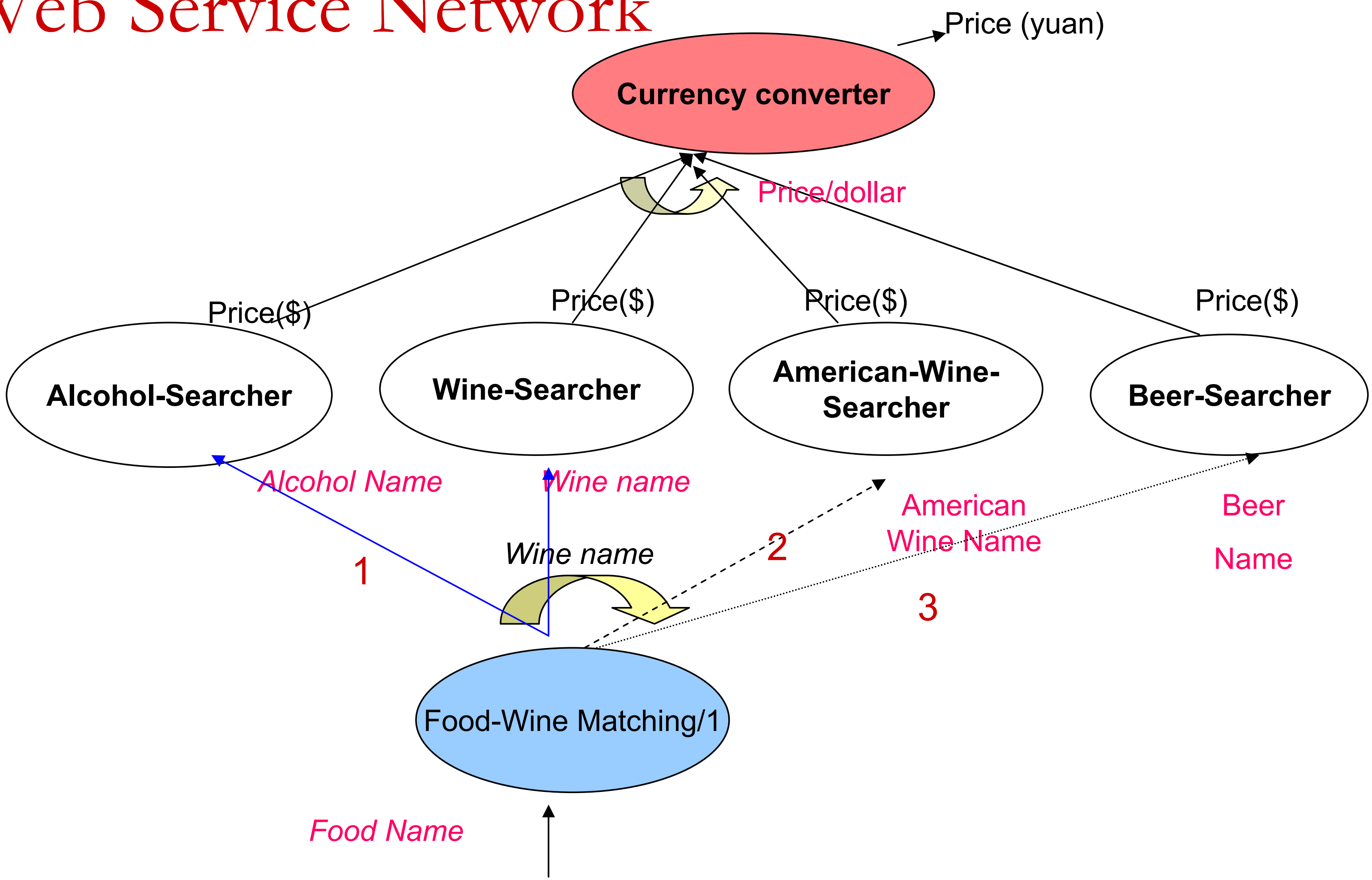
Web Service Ontology and Domain Ontology



Query Format

```
<Query: QueryName> Wine Price</profile: serviceName>

<Query: qualityRating>
<profile: qualityRating rdf:ID="Query-Rating">
<profile: ratingName> average </Query: ratingName>
<profile: rating rdf:resource="owl-q/1.0/Concepts.owl#GoodRating">
 </Query:QualityRating>

<Query: hasInput   rdf:resource="Service-Concept.owl # seafood/Food"/>
<Query: hasOutput  rdf:resource="ServiceConcept.owl # wineName/Wine"/>
<Query: hasOutput  rdf:resource="Service-Concept.owl # winePrice/Wine
            <daml:Restriction daml:onProperty rdf:resource=Franc">
            </daml:Restriction>
```

Web Service Network

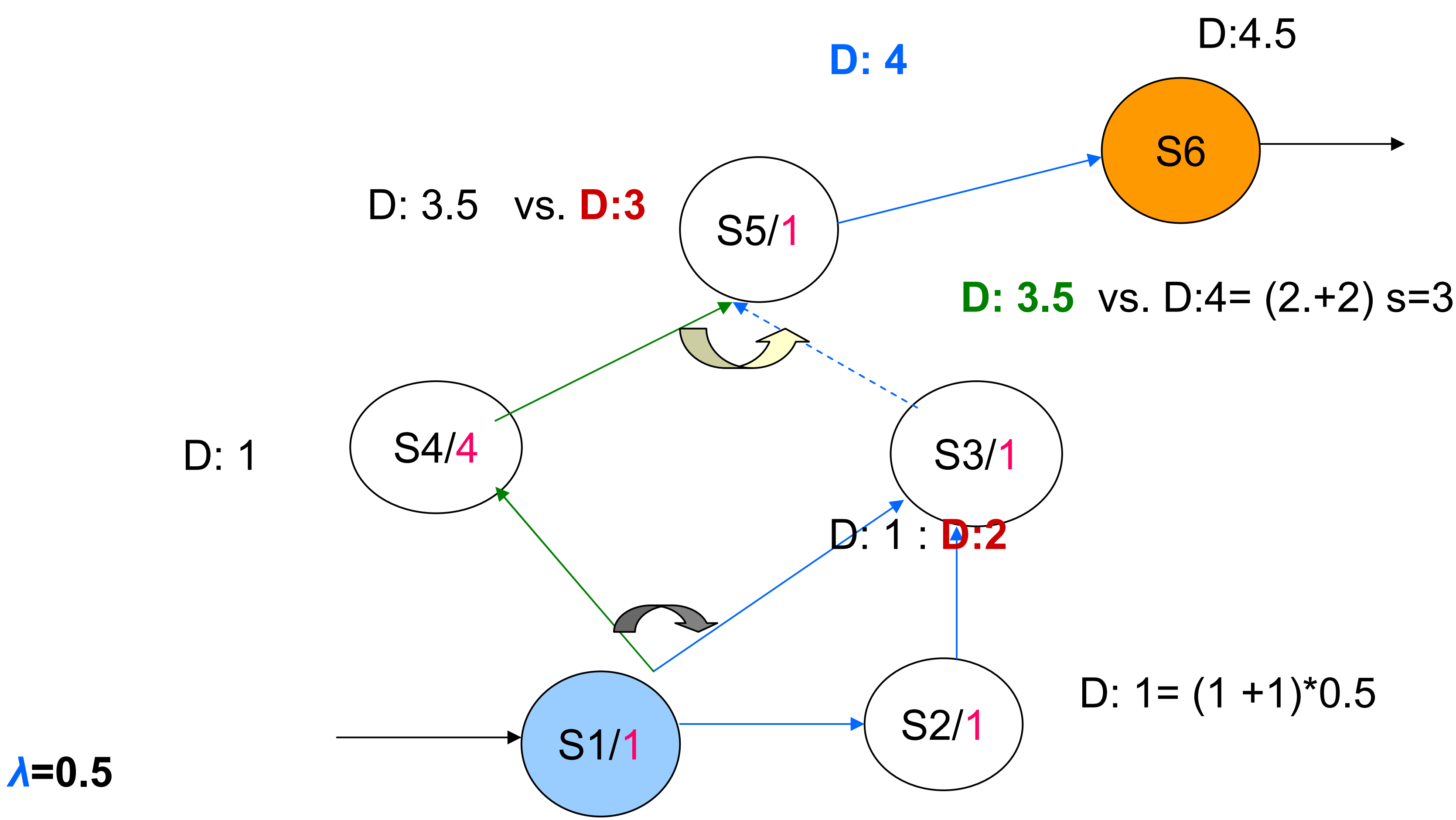


Algorithm

$W = (1-\lambda) * \text{quality rate} + (\lambda) * \text{similarity value}$

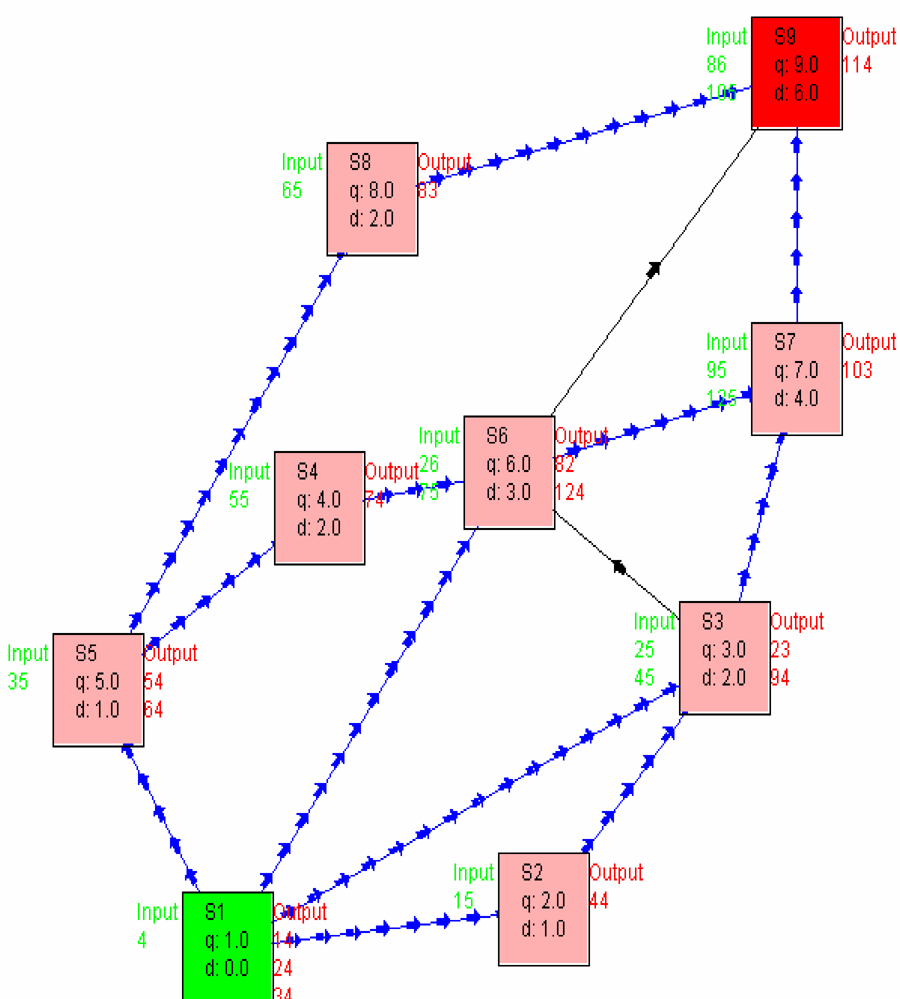
Quality rate can be other QoS measurements. Such as cost of time.

Algorithm

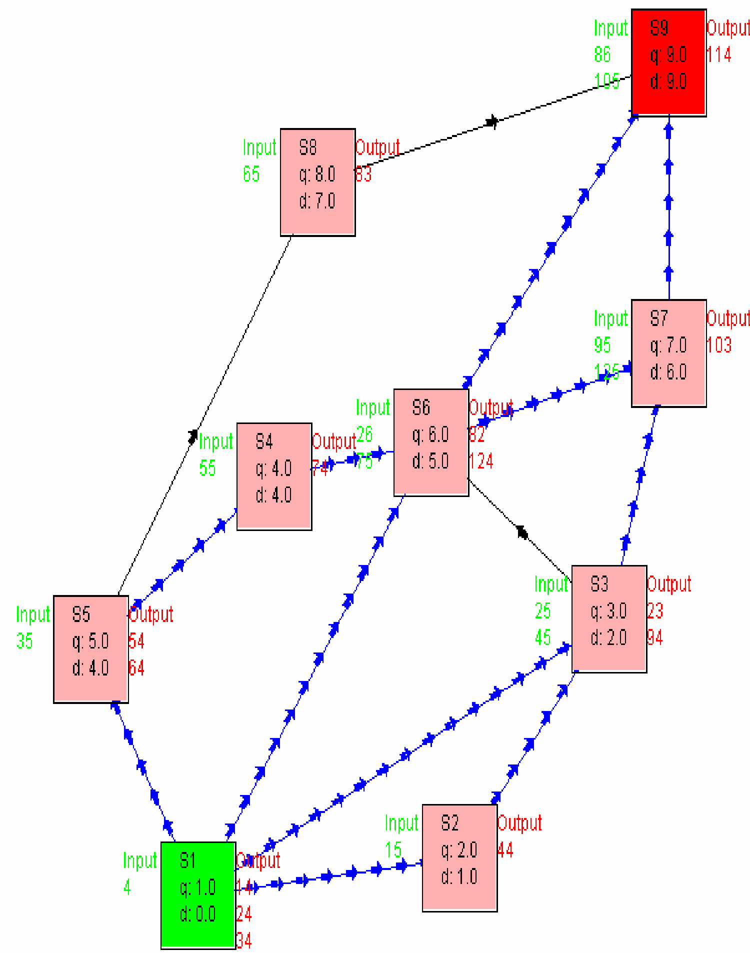


D:4.5
D: 4
S6
D: 3.5 vs. D:3
S5/1
D: 3.5 vs. D:4= (2.+2) s=3
D: 1
S4/4
S3/1
D: 1 : D:2
D: 1= (1 +1)*0.5
S2/1
S1/1
λ=0.5

Example: General Cases

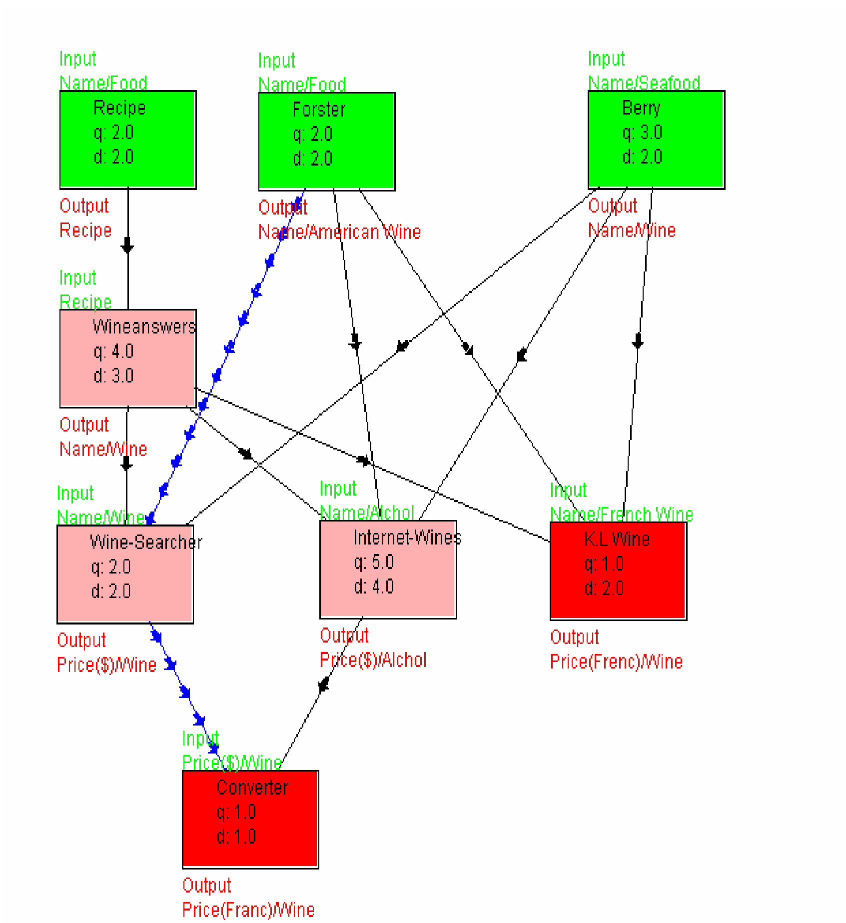


$\lambda = 0.3$

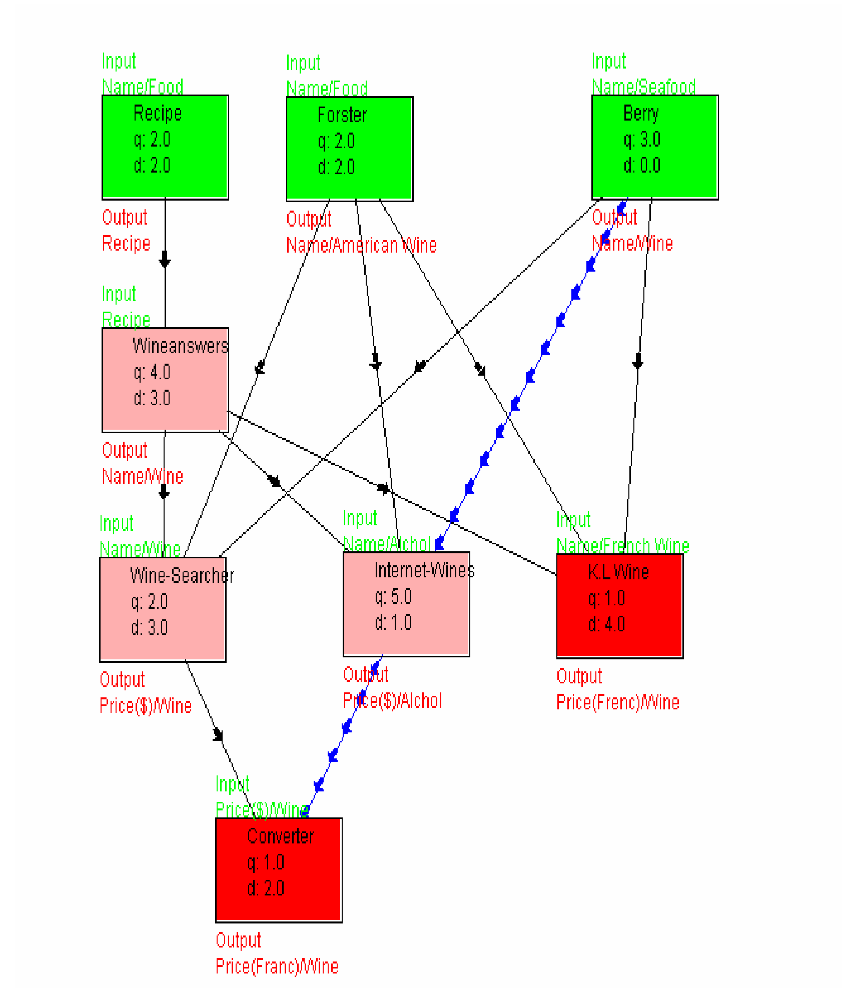


$\lambda = 0.7$

Food-Wine Matching

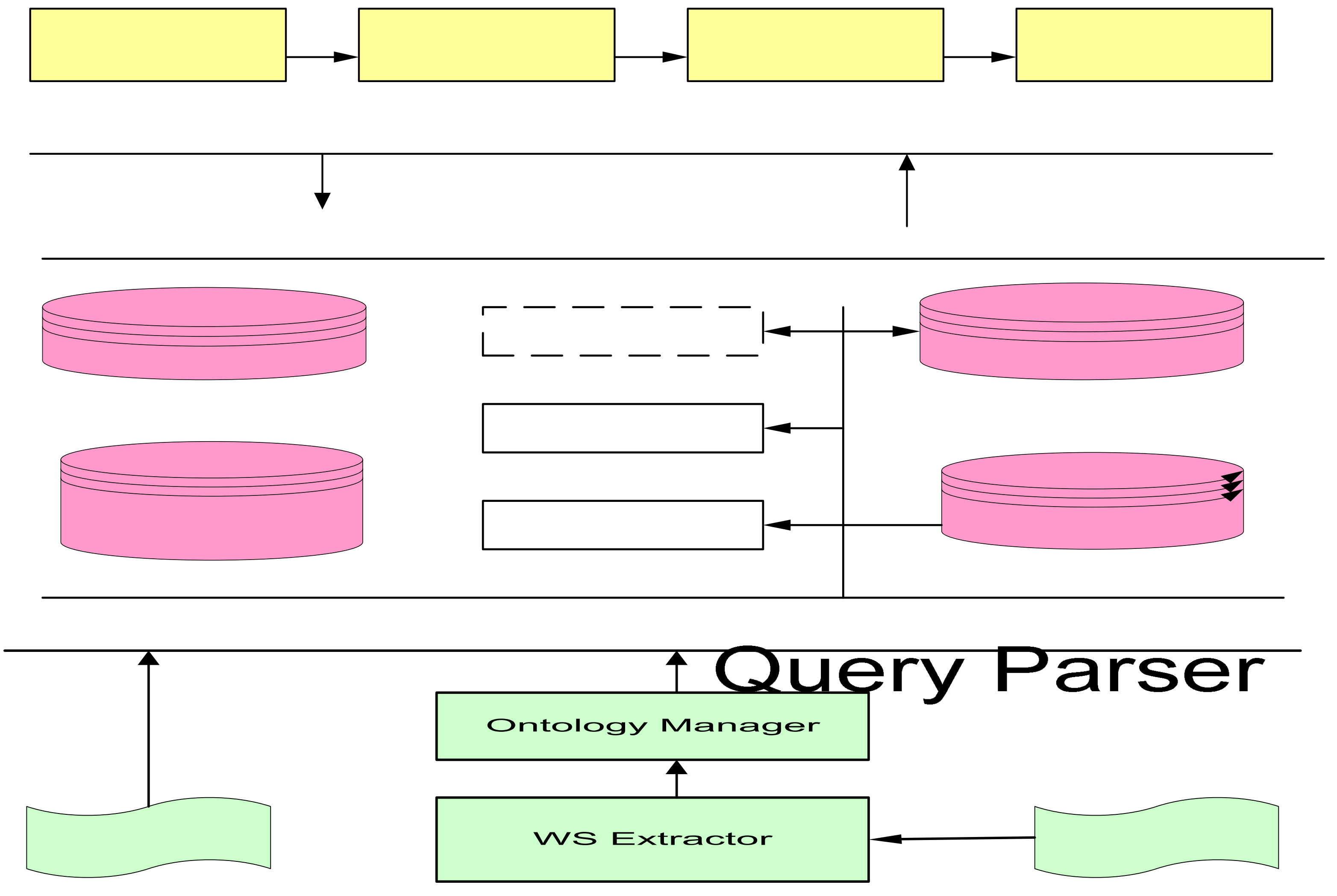


$\lambda = 0.2$

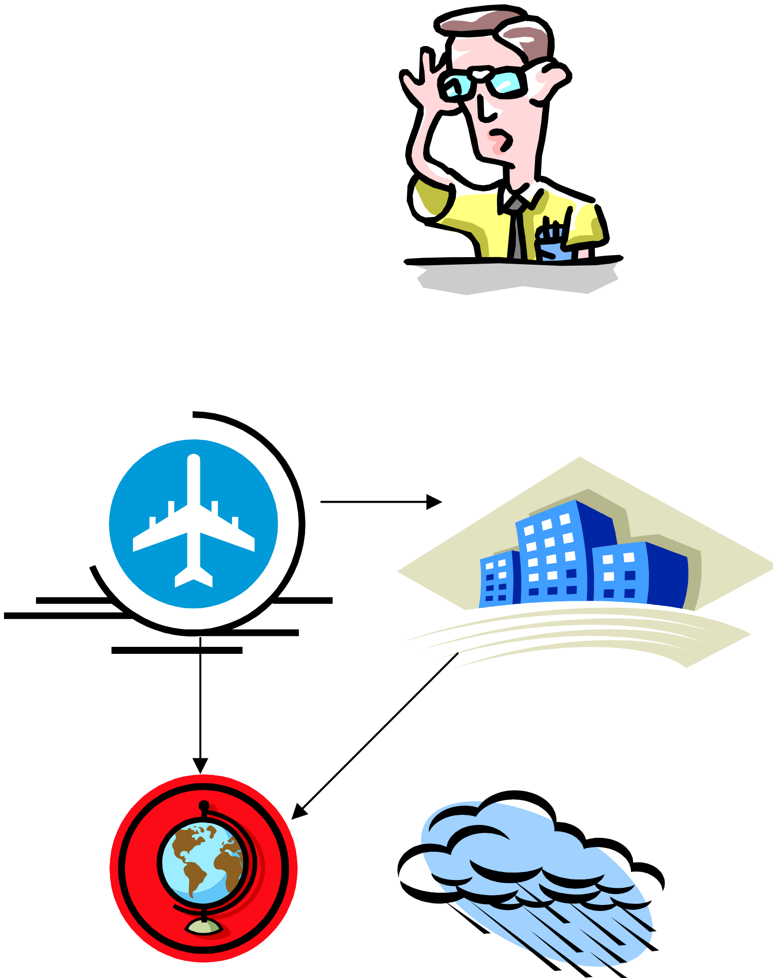


$\lambda = 0.8$

Architecture



Interactive Composition?



- Dynamic binding
- Efficient filtering
- Process Modified in customized Process
- Service instance determined by values produced at runtime.

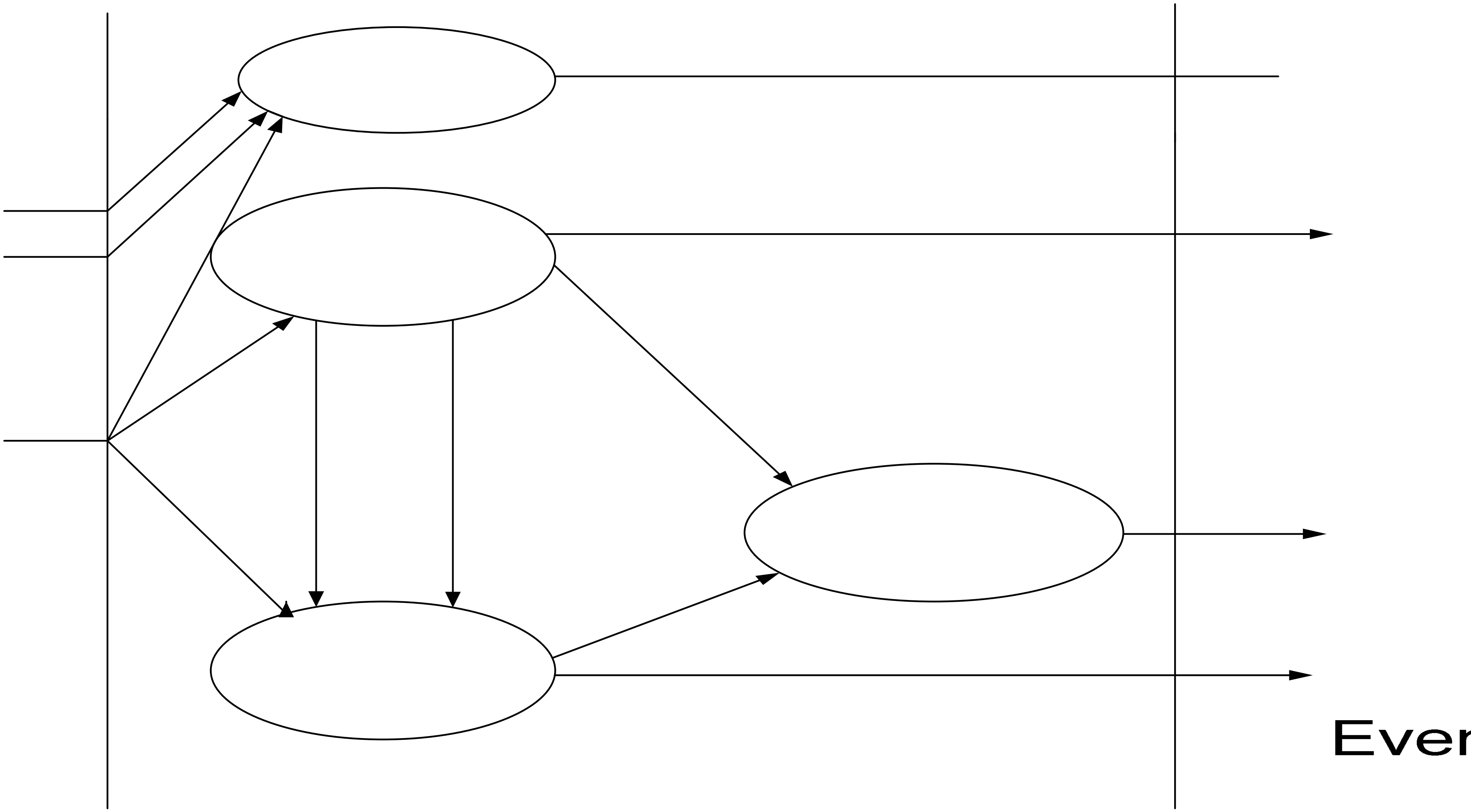
Human-Assisted Composition

- Interface-matching is not enough for complex service
- Consider Quality rate, cost of time, geographic region, user profile and other attributes.
- Service output values (such as price of ticket)
- Template-based composition

Motivating Example

- *Consider a user, who is planning a round trip to London, U.K. from Atlanta, GA from May 1st to May 15th.*
- *Initially, the system displays a travel planner for the composition*

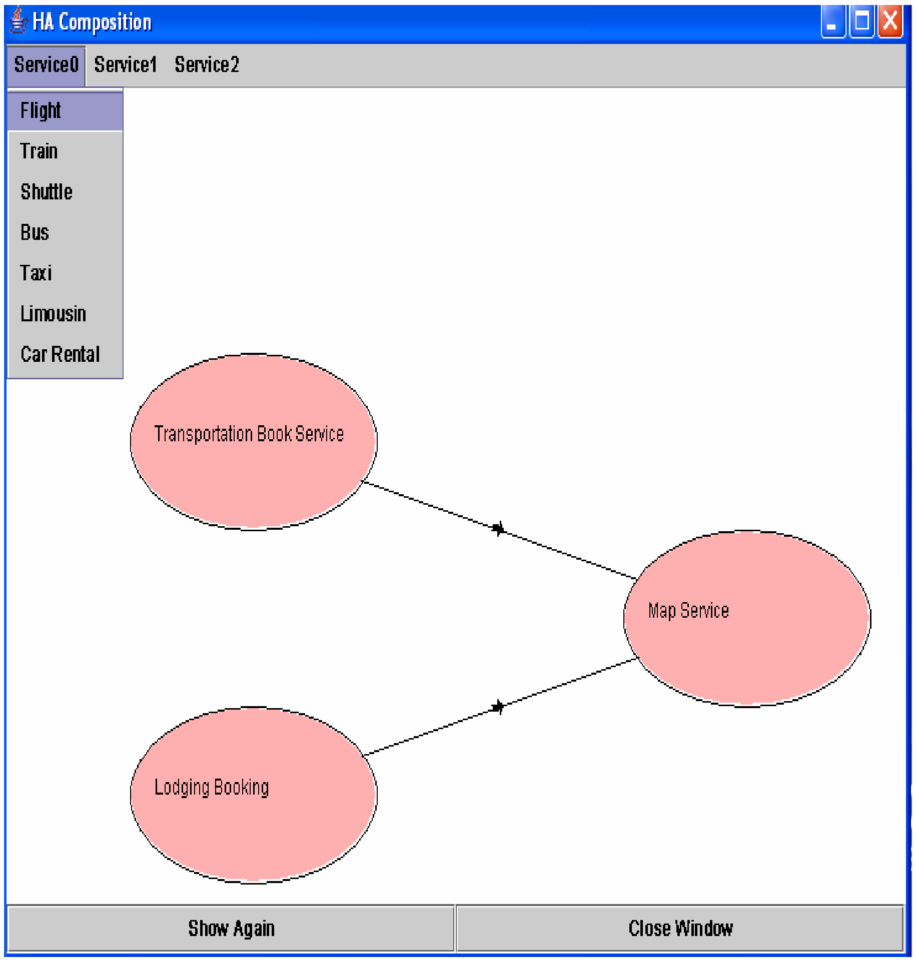
Travel Planner

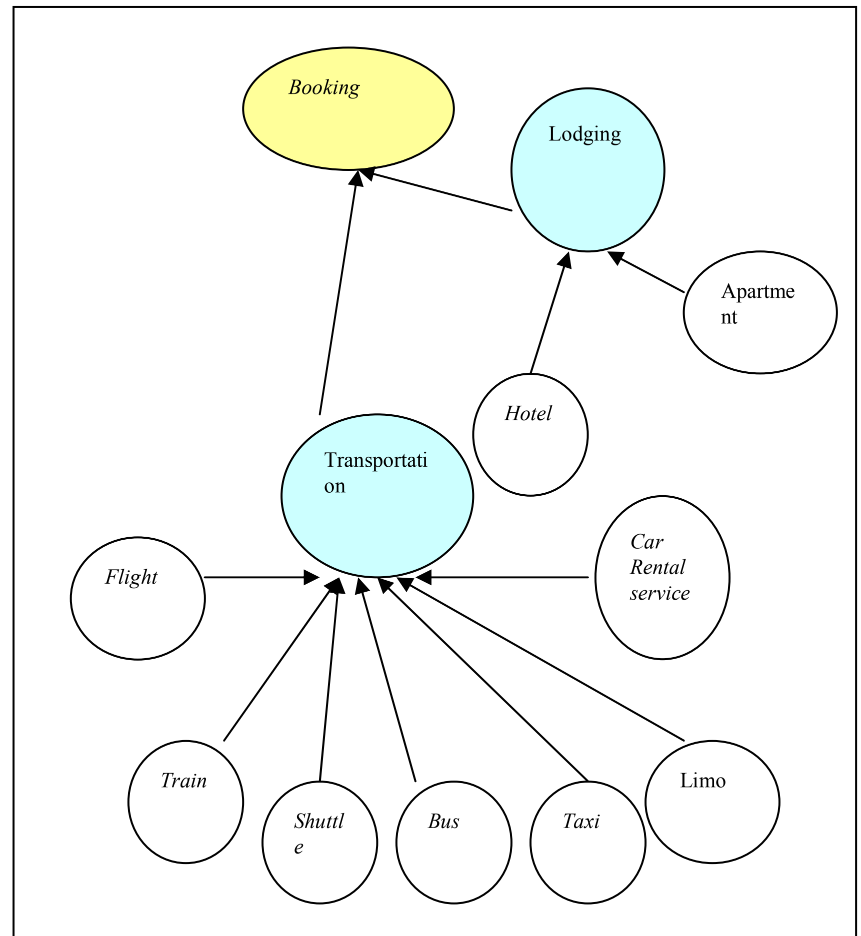


Selection Procedure

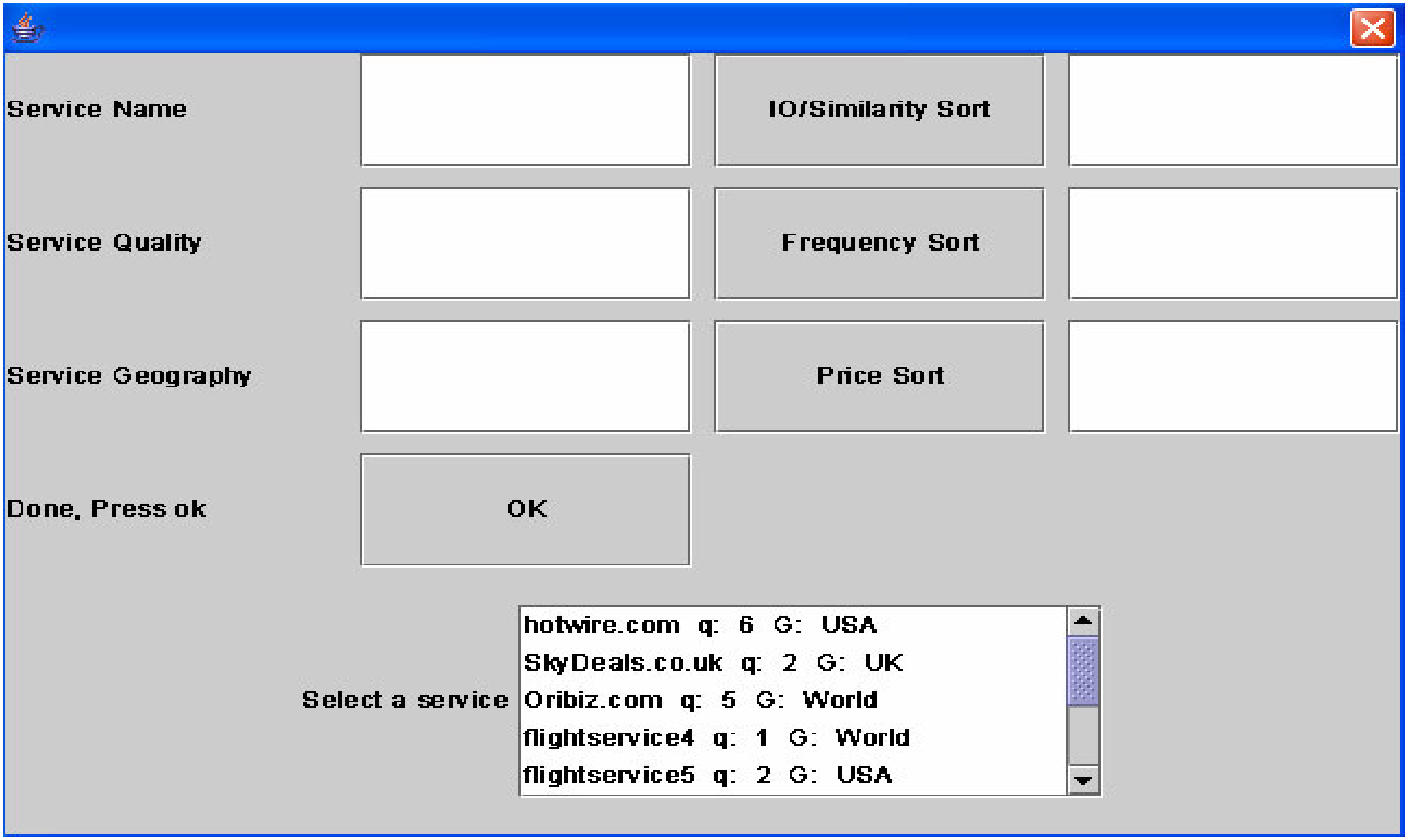
1. Selection for service classes.
   - Select the appropriate subclasses of the services
2. Selection for service instances.
3. Selection for neighboring services.

Step I: Selection For Service Classes





Step 2: Selection for Service Instances

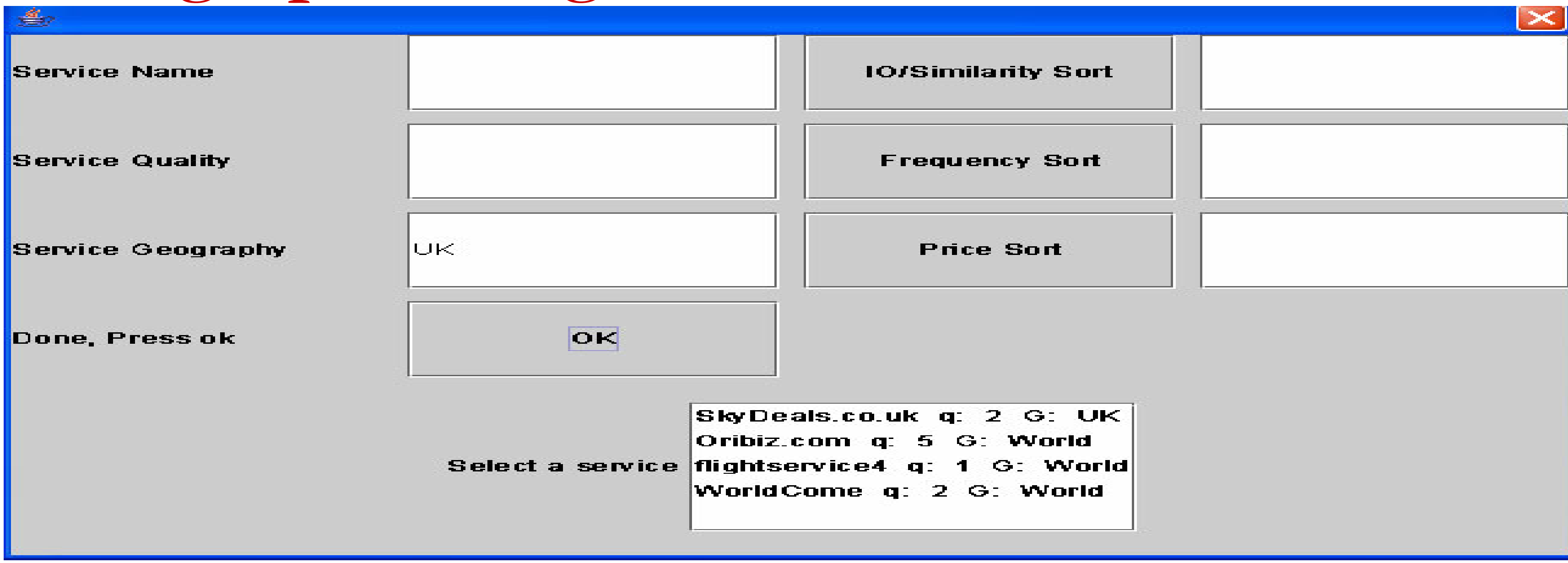


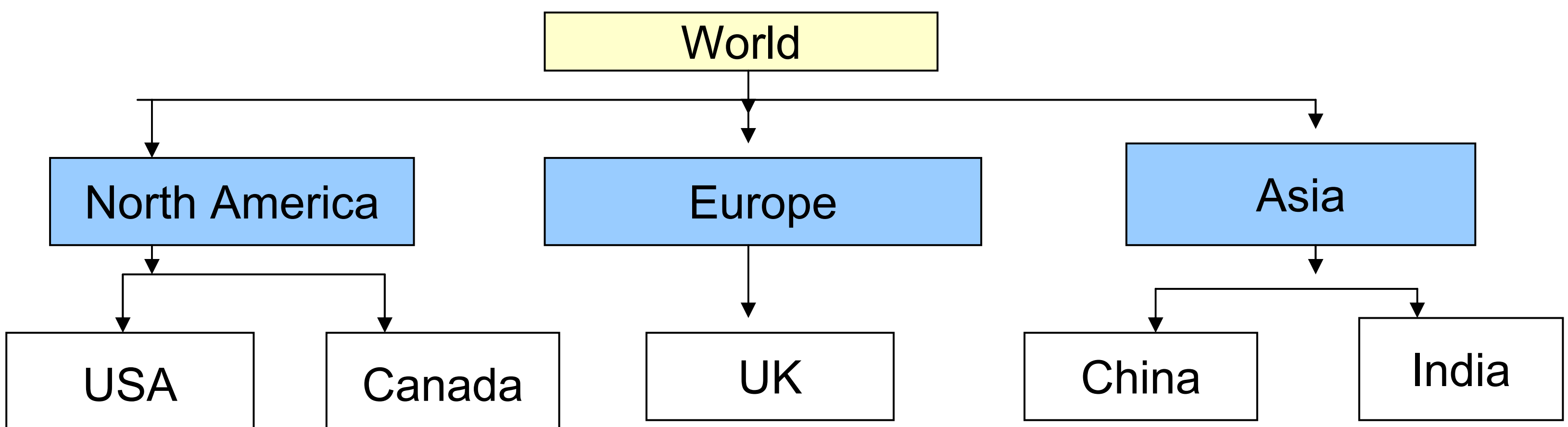
Service Name and Quality Filters

1. Service Name Filter
   - key-word search
   - Multiple Names [METEOR-S (MWSDI)]

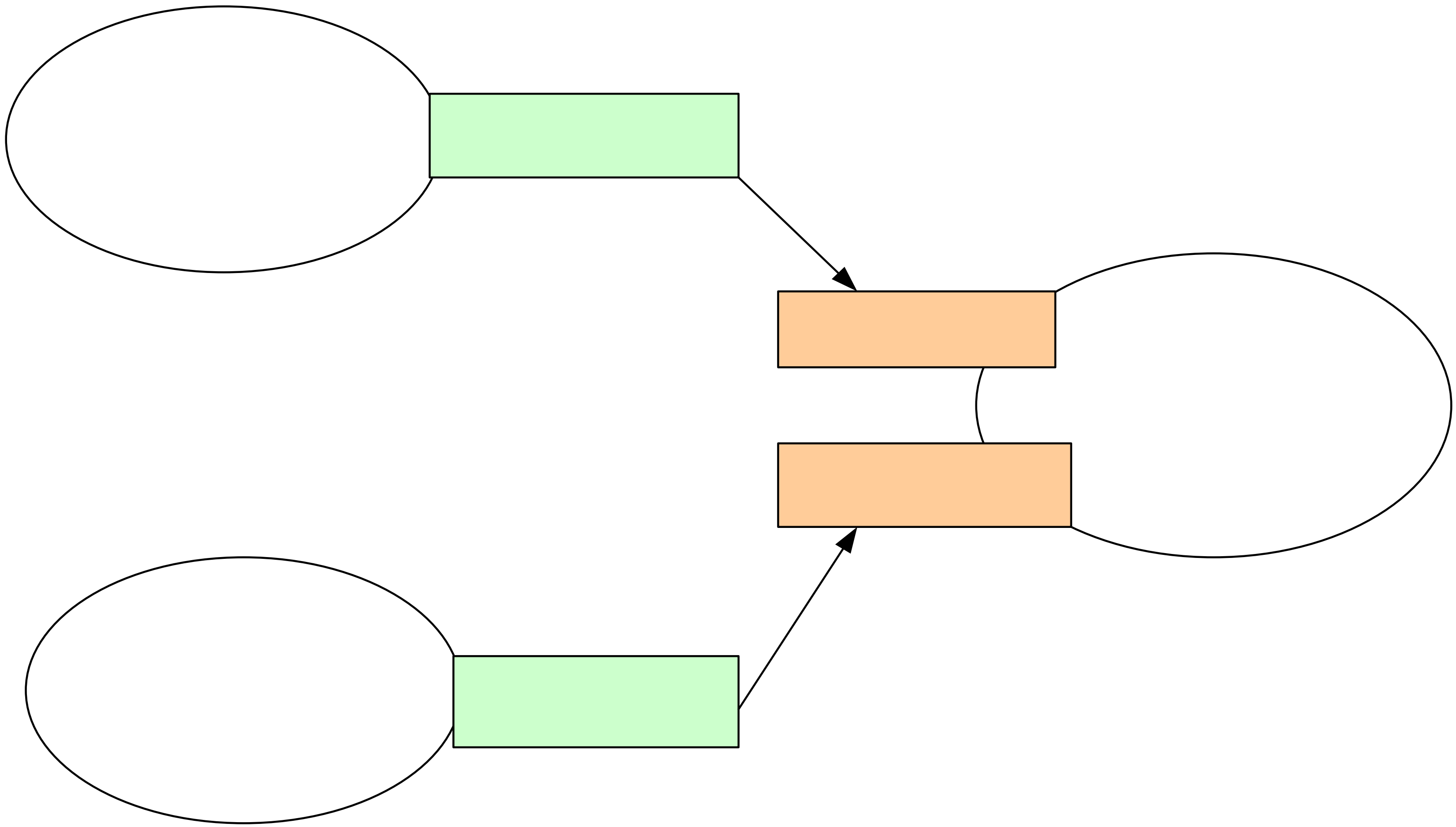
2. Service Quality Rate Filter
   - Quality rate ontology
   - Sorting algorithm
   - Be used individually or with other filters.

Geographic Region Filter

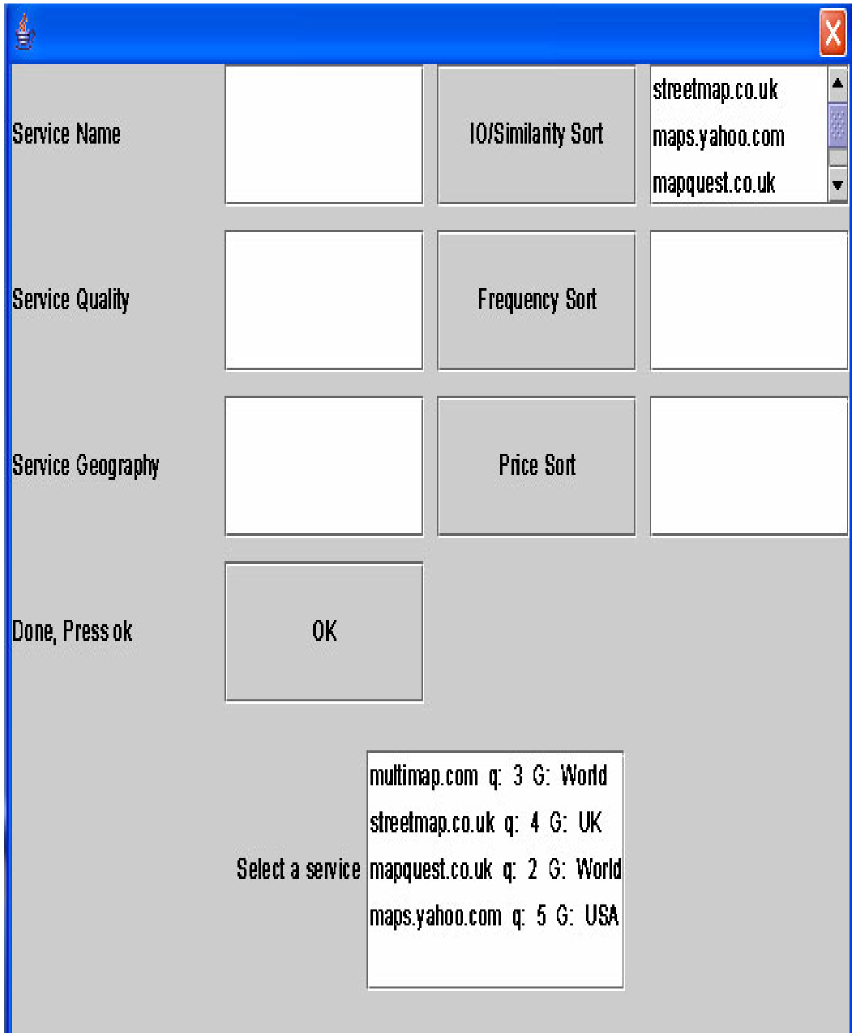


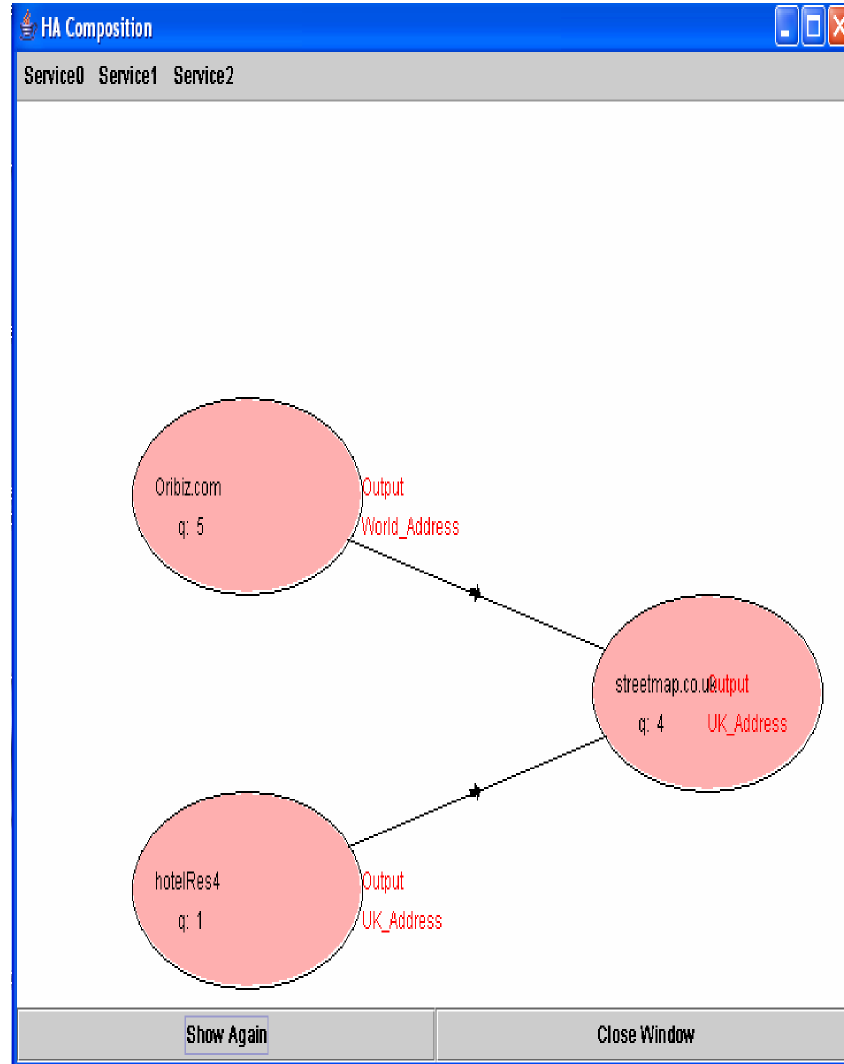


IOPE Similarity Filter (I)



IOPE Similarity Filter (II)

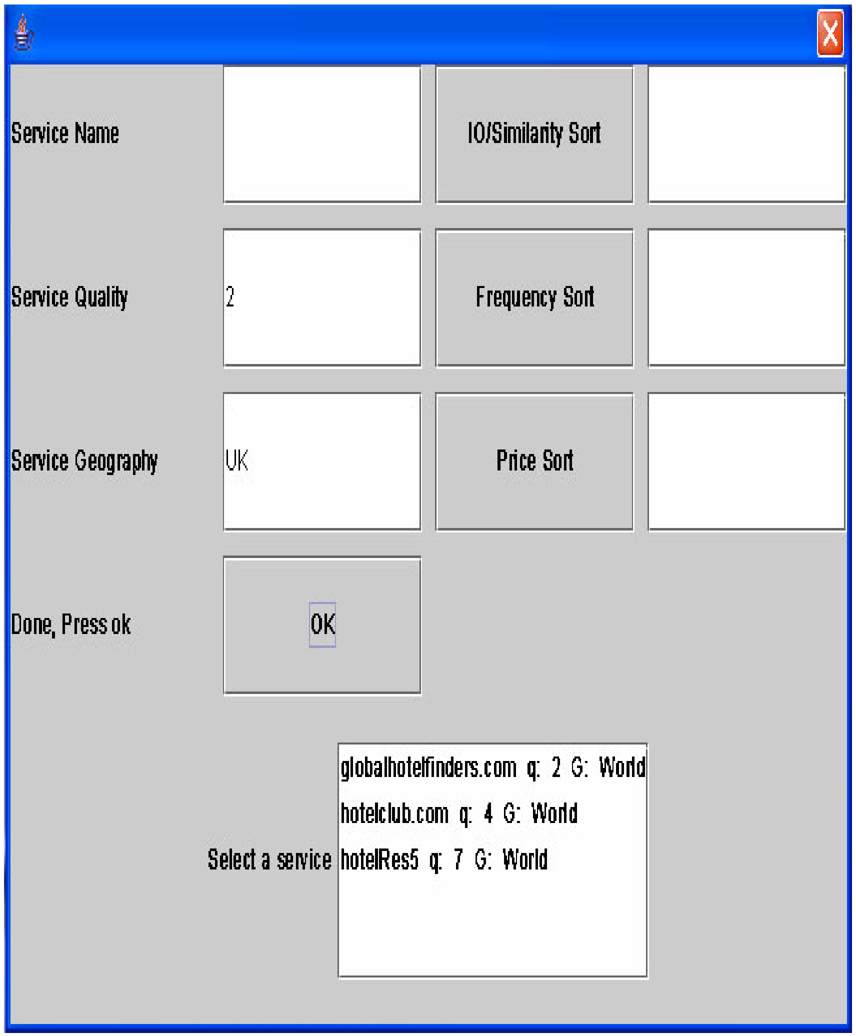


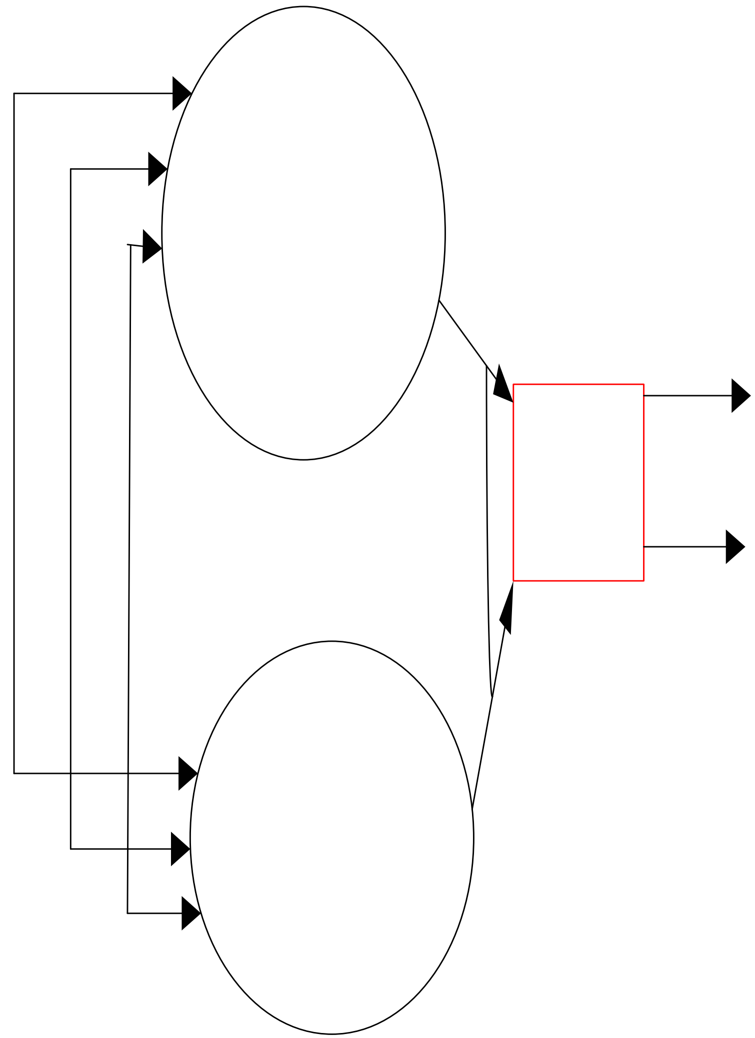


Personal Profile Filter

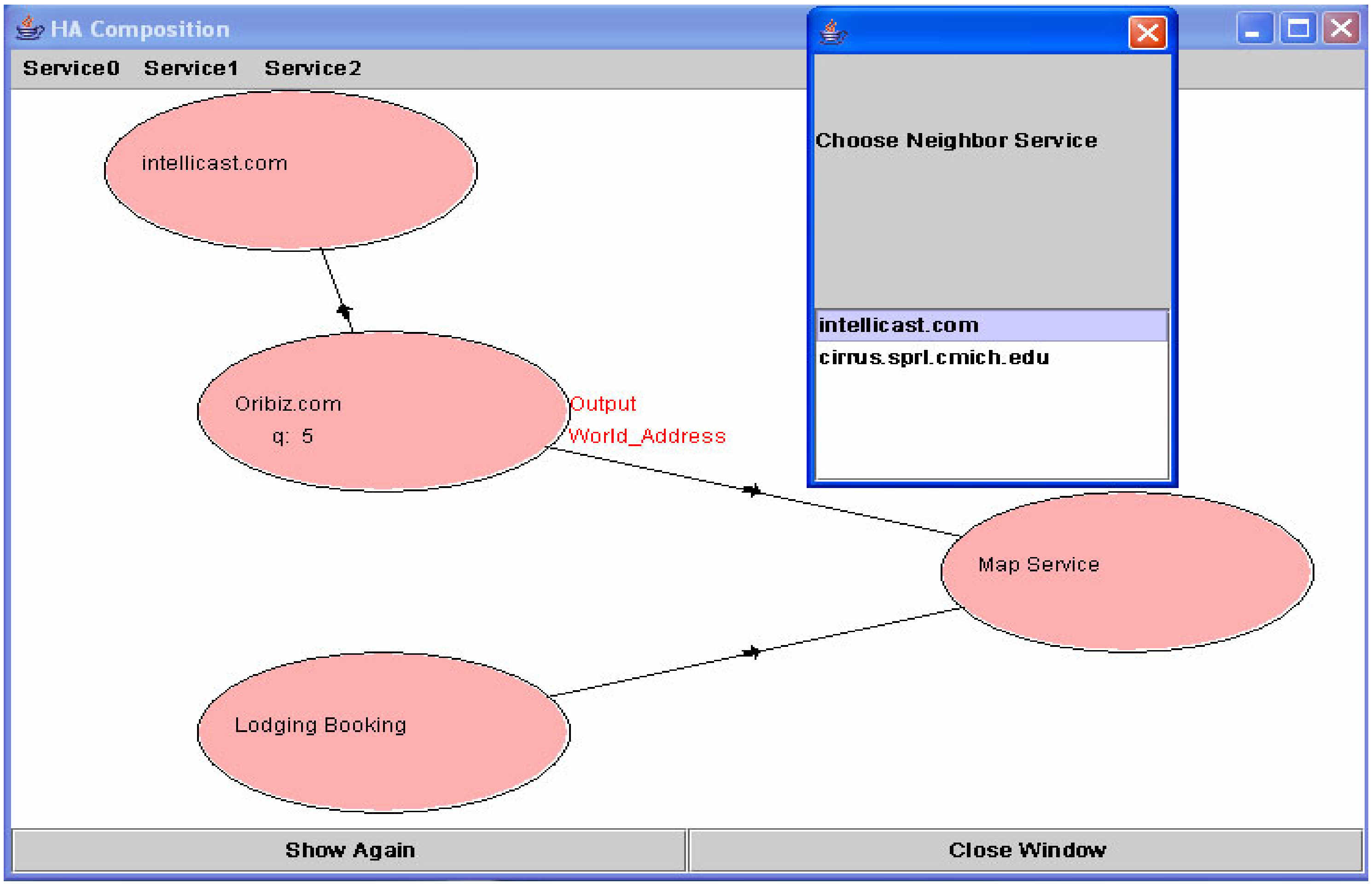
- The personal profile records the history of service instances used involving usage frequency by the user.
- We assume that the service with highest usage frequency is most likely to be selected in the future.

Price Filter

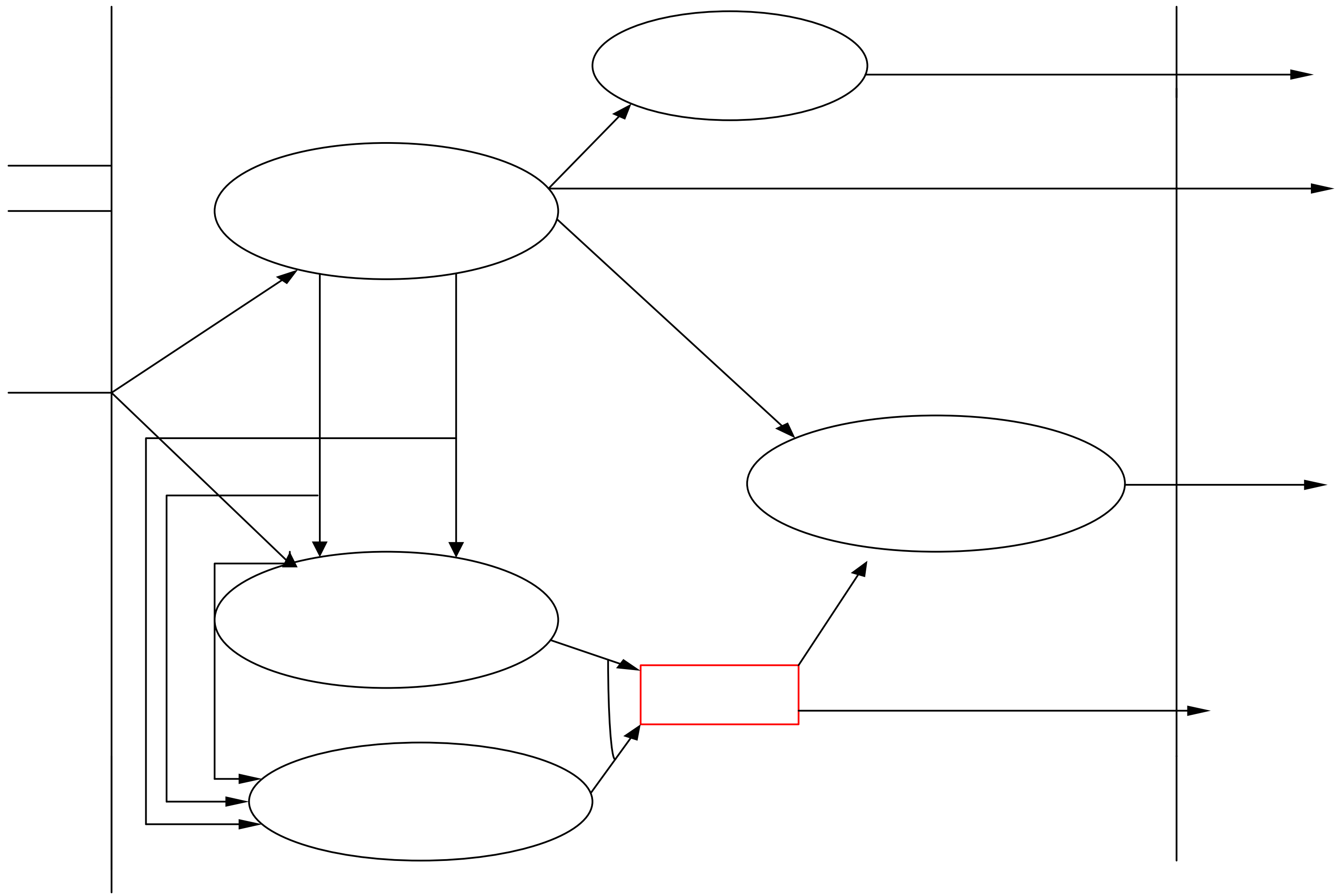




Step 3:Selections for Neighboring Services



Trip Composition Service Graph



Arrival Date

Semantics in HA

- Web Service Ontology for Web Service class selection
- Domain Ontology for Web Services instance selection, (quality, geographic region, IO matching)
- Web Service Network for Neighboring Service Selections (functionality Semantics)

Contributions

- Explicit ontological service descriptions.
- Found optimal composition in a flexible way (Qos)
- Developed filters to help users to make better service selection decision in composition.

- Automatic Composition of Semantic Web Services, R. Zhang, B. Arpinar, and B. Aleman-Meza, Intl. Web Services Conference, Las Vegas NV, 2003.
- Ontology-Driven Web Services Composition, B. Arpinar, R. Zhang, B. Aleman-Meza, and A. Maduko, IEEE Conference on E-Commerce Technology (CEC 2004), San Diego, California, July 6-9, 2004 (accepted).

Future Work: Functionality-based Composition

- Composition based on their internal computations when their profiles may not convey adequate semantics to differentiate them.
- Some thoughts:
  - Black-box approach:
    - Exploit pre- and post-conditions in composition
  - White-box approach:
    - Process ontology
    - State transformations (e.g., Petri nets)
    - Process Query Language Klein

Reference

- [Srivastava03] B. Srivastava, J.Koehler. Web Service Composition –current solutions and open problem. *Icaps 2003 Workshop on Planning for Web Services*
- Adapting Golog for Programming the Semantic Web. S. McIlraith, T.C. Son.
- [Klein01]M. Klein, and A. Bernstein. Searching for Services on the Semantic Web Using Process Ontologies*, International Semantic Web Working Symposium, August 2001.*

Questions?

Thanks