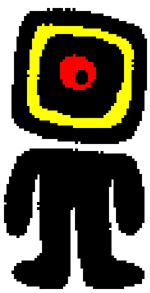
Semantic Brokering over Dynamic Heterogeneous Web Resources



Anne H. H. Ngu
Department of Computer Science
Southwest Texas State University
November 2002

Overview

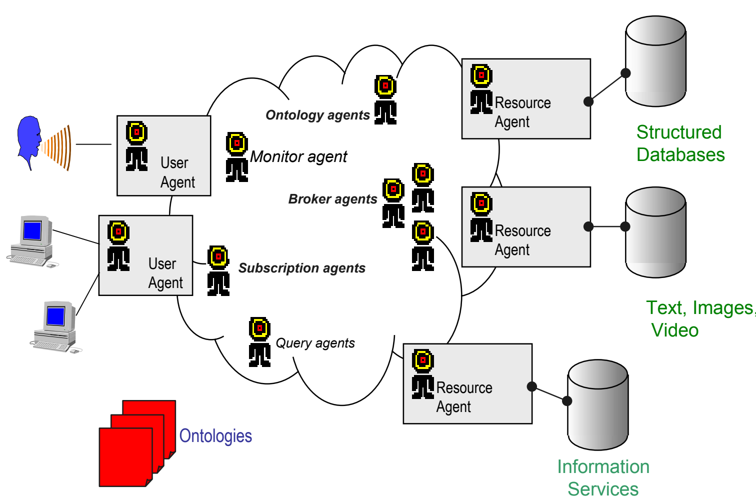
- Objectives of data integration in InfoSleuth system.
- InfoSleuth architecture
- Role of brokering and ontology in data integration
- Multibrokering design and implementation
- Performance evaluation of multibrokering system
- Dynamic integration and coordination of services

Goals of InfoSleuth

- Development of technologies and tools to support concept-based access to information sources in a dynamically changing web environment through mediated interoperation of agents.

•It allows concept-based search, retrieval and fusion of related information from changing set of web resources.
•It monitors dynamic information sources for relevant changes and aggregates changes to multiple level of abstraction and notification.
•It provides for easy evolution by allowing plug-in of new users, new resources and new services.

InfoSleuth Architecture

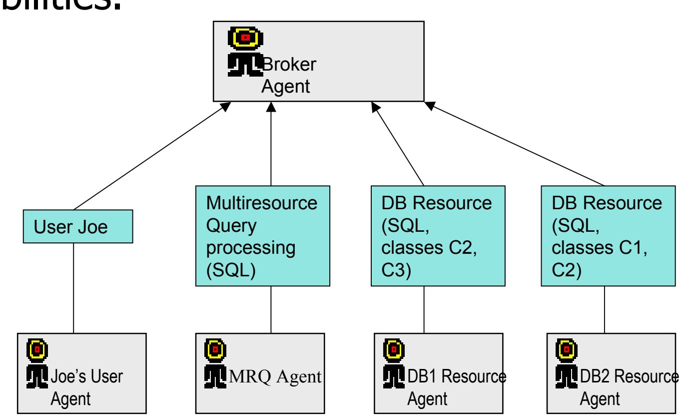


InfoSleuth Architecture (Cont.)

- Core agents collaborate to service requests over a common ontology.
- Resource agents serve as mediators to external information sources such as structured DBMS, semi-structured web pages, multimedia sources etc.
- User agents act as proxies for individual users or group of users.

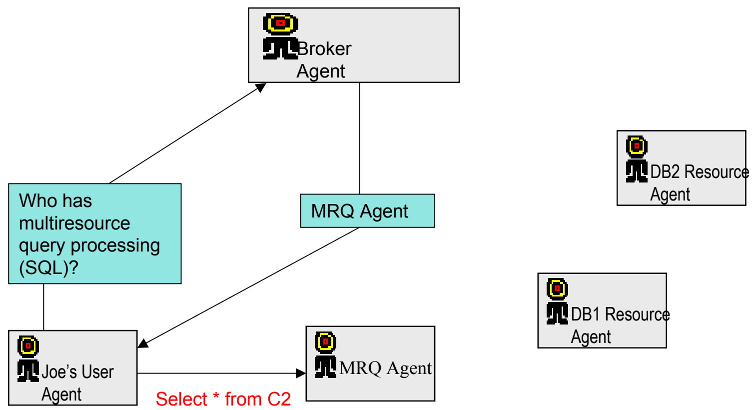
Overview of how agents collaborate

- As each agent comes online, it advertises its capabilities.



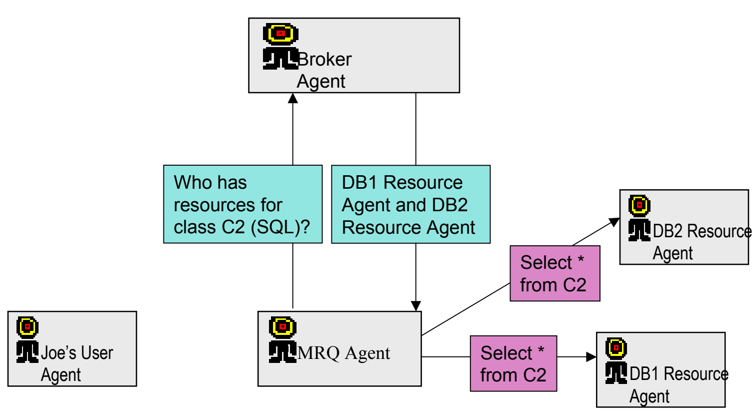
Overview of how agents collaborate (Cont.)

- User Joe submits SQL query select * from C2 to his user agent



Overview of how agents collaborate (Cont.)

- MRQ agent looking for resource agents that can answer an SQL query involving class C2.



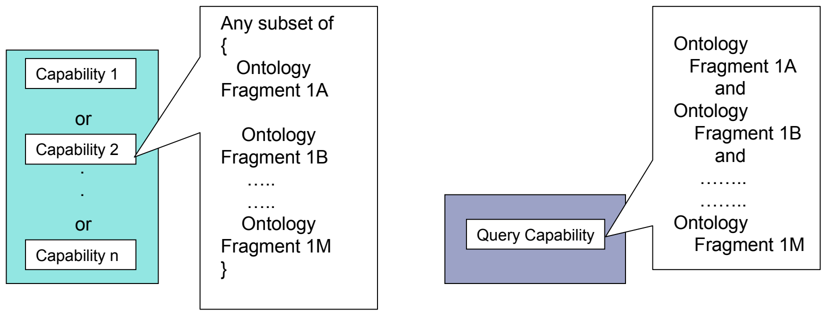
Broker Agent Functions

- Repository
  - accepts and stores agent advertisments
  - maintains the state of the system, periodically prunning non responding agents
- Matchmaker
  - reasons over agent capabilities and their information contents
  - recommends only potentially relevant agents for a task

Focused Ontology and Ontology Fragments

- One single global ontology
  - ✓ relationships among different aspects of agent capabilities can be represented
  - \- difficult to manage inter-domain relationships and to add new ontological concepts
- Multiple, focused ontologies
  - ✓ adding a new ontology is easy
  - ✓ capabilities of agents can be composed easily in terms of ontology fragments

Agent Capabilities, Advertisements and Queries



Agent Capabilities and Advertisments **Agent Capabilities and Queries**

Example of an advertisement

```
<advertisement>
<capability NAME-"ResourceAgent5Cap">
  <ontology_fragment NAME="_infoSleuth" VERSION="1.0"
    <class NAME="agent"
      <slot NAME="agent address" VALUE="tcp:research.telcordia.com:7000"</slot>
      <slot NAME="agent name" VALUE="ResourceAgent5"></slot>
      <slot NAME="type" VALUE="resourceagent"></slot>
    </class>
  </ontology_fragment>
  <ontology_fragment NAME="_conversation" VERSION="1.0">
    ….
  </ontology_fragment>
  <ontology_fragment NAME="sql" VERSION="1.0">
    <class NAME="select-statement"></class>
  <ontology_fragment>
  <ontology_fragment NAME="healthcare" VERSION="1.0">
    <class NAME="diagnosis">
       <slot NAME="diagnosis-code"> </slot> </class>
    <class NAME="patient">
        <slot NAME="patient-age"></slot>
           <constraint> <set_interval> MIN_VALUE="43" MAX_VALUE="75" </set_interval></constraint>
           <key NAME="patient-id"></key>
     </class>
      …
</capability> </advertisement>
```

Example of a Query

```
<query>
 <capability NAME="_generic_query_capability">
   <ontology_fragment NAME="_infosleuth" RETURN_CLASSES="false">
     <class NAME="agent" RETURN_KEYS="false" RETURN_SLOTS="false" SLOT_SEMANTICS="all"
       <slot NAME="agent name" RETURN_CONSTRAINTS="true"></slot>
       <slot NAME="agent address" RETURN_CONSTRAINTS="true"></slot>
         <constraint_conjunct> </constraint_disjunct>
           <slot NAME="type" VALUE="resourceagent"></slot>
        </constraint_disjunct></constraint_conjunct></class>
        …
   <ontology_fragment NAME="healthcare" RETURN_CLASSES="true" CLASS_SEMANTICS="any">
     <class NAME="patient" RETURN_KEYS="true" RETURN_SLOTS="true" SLOT_SEMANTICS="any">
       <constraint_conjunct><slot NAME="patient-age" RETURN_CONSTRAINTS="true">
         <set_interval> MIN_VALUE="45" MAX_VALUE="65" </set_interval>
       </constraint_conjunct>
      </class>
      ……
 </query>
```

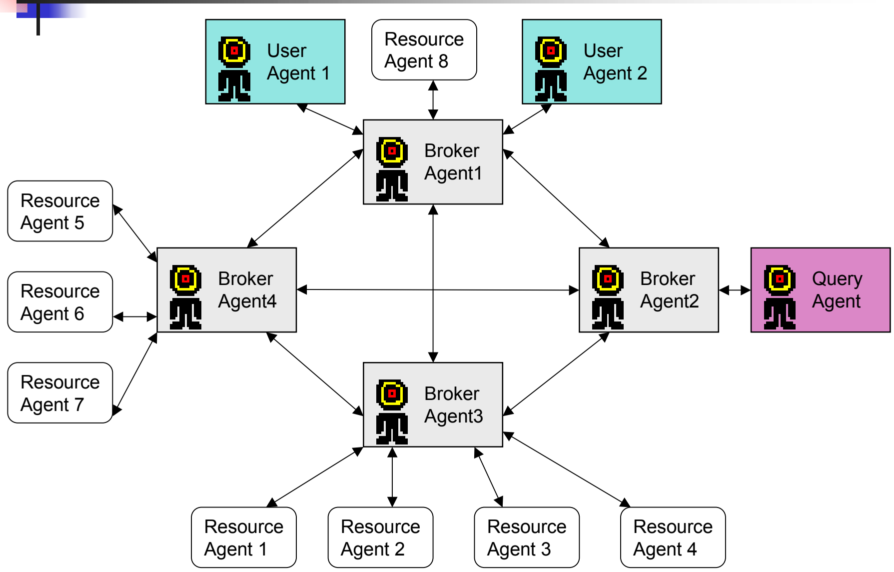

Problems with Single Broker Architecture

- A single point of failure
- Represent a hard limit to scalability
- The reasoning engine degrades as the broker's repository grows bigger

Principle for Scalable Multibrokering

- Peer-to-Peer Architecture (not hierarchical)
  - brokers may freely advertise or unadvertise to any broker
- Non-broker agents must advertise to more than one broker
  - robustness increases if agents advertise redundantly to several brokers
- Brokers should specialize
  - helps in limiting search space when broker specialities are known and advertised.

Multibroker Architecture



Implementation of Multibrokering

- Collaborative Reasoning
  - how to ensure that brokers process queries collaboratively and thoroughly
- Integrating new broker and non-broker agents
  - how new broker find other brokers
  - how non-broker agents find the brokers
- Maintaining Connectivity
  - how to ensure that all brokers and agents remain interconnected

Collaborative Reasoning

- Each broker can forward queries to other brokers that may have other matching agents.
- Inter-broker search is initiated based on the nature of the request and a search policy.

Recommend-one - brokers are searched one by one in a breadth-first manner until a match is found.
Recommend-all - brokers are searched in parallel until all accessible brokers have been queried.
Hop count - defines how many hops should be traversed for a given query.
broker trail - prevents cyclical propagation of search.

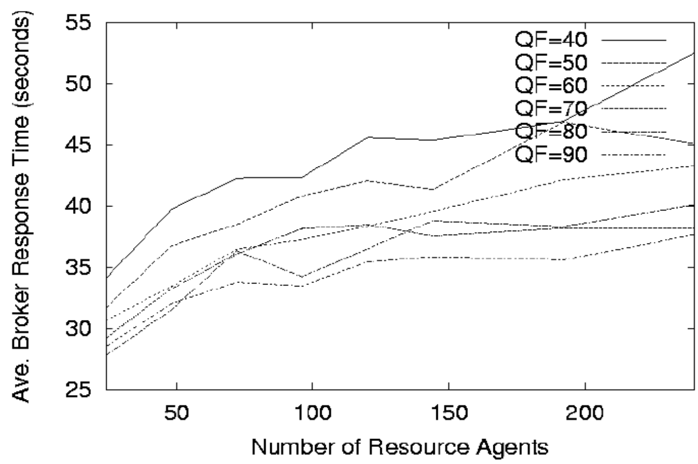
Integrating new agents and brokers

- New broker is configured with a list of other brokers, or a well known port it should advertised to.
- New broker advertises its location information and capabilities to other brokers.
- Non-broker agent is configured with a list of known brokers to connect to on startup.
- Non-broker agent can re-configure to a different broker or a different set of brokers later by monitoring quality of service of current brokers.

Maintaining Connectivity

- Redundant advertising
  - all agents keep a known-broker-list and a connected-broker-list
  - each agent or broker advertised to the known-broker-list, until the connect-broker-list reaches its max configured parameter
- Robust connectivity
  - broker periodically pings all agents
  - agent periodically ping all its connected brokers.
  - re-advertise when the connected-broker-list is less than the max configured number.

Scalability Experiments



Experimental Query Streams

QUERY TEST STREAM	NUMBER OF RESOURCE AGENTS
SA (single agent)	1
DA (double agent)	2
4A (four agent)	4
VF (vertical fragmentation)	4
CH (class hierarchy)	4
FH(fragmentation & class hierarchy)	4

Experimental configurations

EXPERIMENT	4A	DA	SA	VF	FH	CH	# RESOURCE AGENTS
A	X						4
B	X	X	X				4
C	X	X	X	X			8
D	X	X	X	X		X	12
E	X	X	X	X	X	X	16
F	X	X	X	X	X	X	16

Experiment F is used to check the effect of broker specialization. Thus resource agents that pertain to a particular query stream are kept with the same broker

Experimental Results

- In an underloaded system, a single broker system has a slightly better response time than a multi-broker system.
  - 1:1.1
- In an overloaded system, a multi-broker system has an improved response time. As the load grows, the difference is significant
  - 1:0.3
- Specialized brokers out perform replicated brokers
- Simulation experiments were also carried out which further confirmed the scalability of the multibrokering system.

Good brokering principles

- Brokering should encompass both syntactic and semantic properties of services.
- Common ontology need to be established for semantic brokering. The focused ontologies approach allows different aspects of agent functionality to be specified and composed.
- Multibrokering enables scalable multi-agents system to be built
- Principles of robust multibrokering and implementation issues:
  - How brokers are connected
  - How brokers discover other brokers
  - How agents discover other agents
  - When to initiate inter-broker search
  - How to maintain connectivity

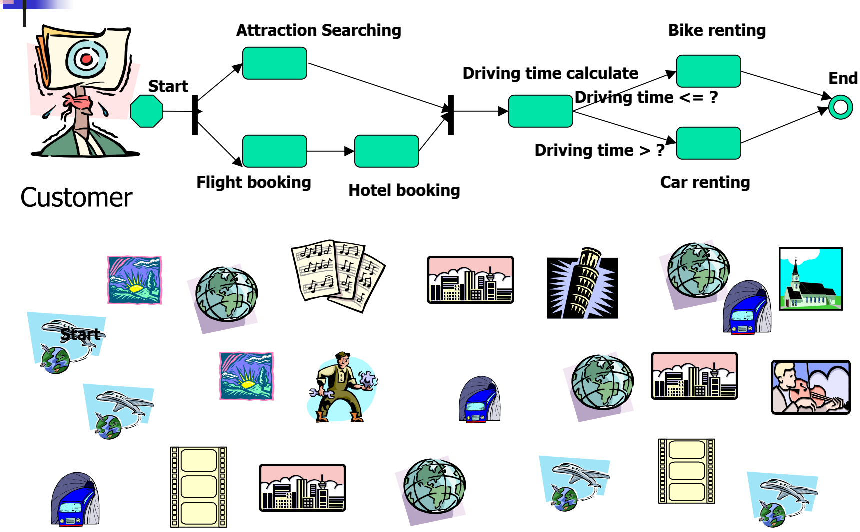
Related Work

- Multidatabases approach:
  - SIMS (Ariadne) at ISI
  - TSIMMIS at Stanford
  - Information Manifold at ATT labs
  - DISCO at INRIA
- Component-based approach:
  - CORBA trading object service
- Other agent based approach:
  - RETSINA at CMU
  - COOL at Toronto University

Agent-approach towards Integration of Services

- New on-line economy requires the ability to efficiently and effectively share business processes and data across the Web and across organization boundaries.
- Multi-agent system has shown to be a viable technology for data integration.
- However, there is a need to move from data to process or service integration.

Example of service integration



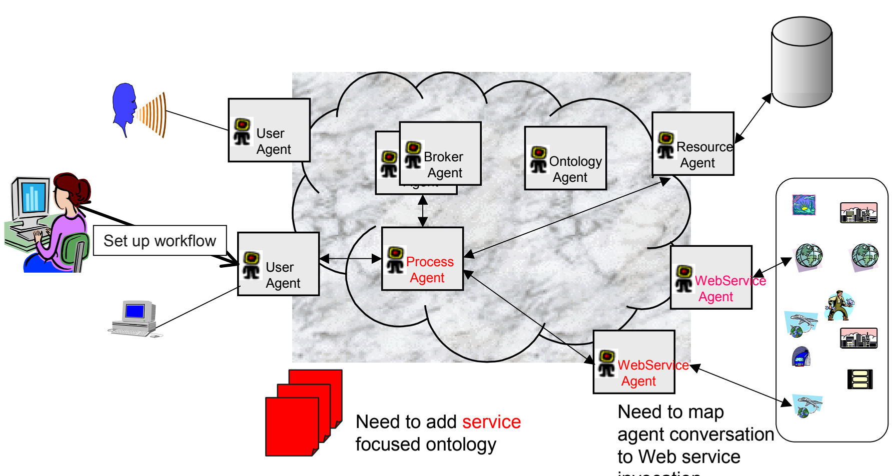
Services are accessible on the web

From Data to Service Discovery

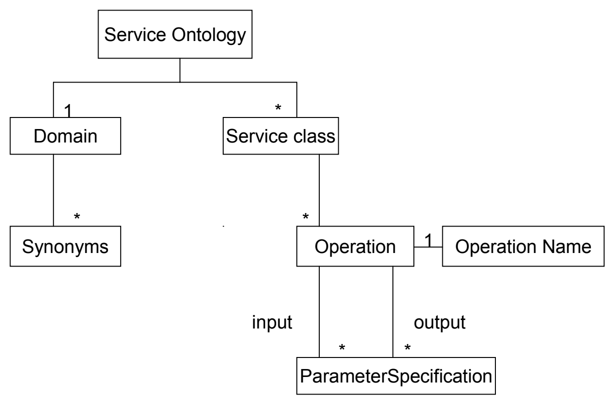
- Service ontology (e.g. WSDL, DAML+S)
  - defines the basic concepts and terminologies which will be used by all the participants in a specific domain
- Service registering/advertising (e.g. UDDI, portal, advertising)
  - a tool for service providers to register their services using a consistent ontology
- Service Selection based on:
  - semantic-brokering
  - negotiation
  - auction
- WebService Agents (service interface and proxy)

Agent-based Approach to Dynamic Composition of Services

User can submit a workflow through the User Agent which will first ask the broker agent for a process agent. The process agent upon receiving the workflow definition will parse it and for each task query the broker for a suitable service agent to execute the task.



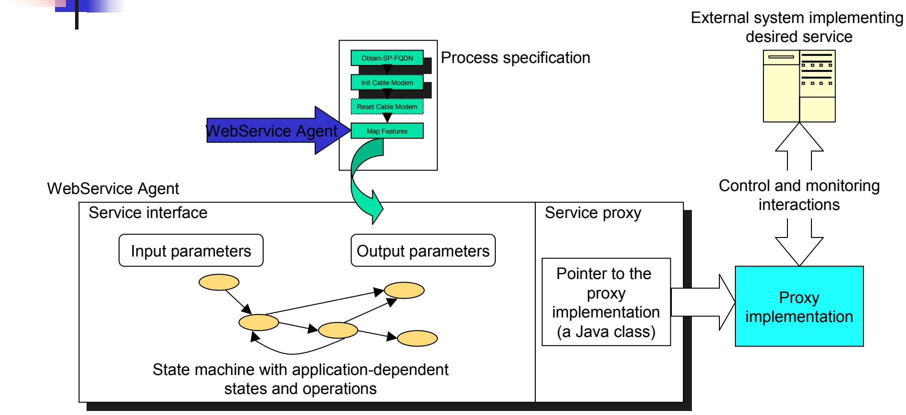
UML class diagram for service ontology



Example of a Service Ontology

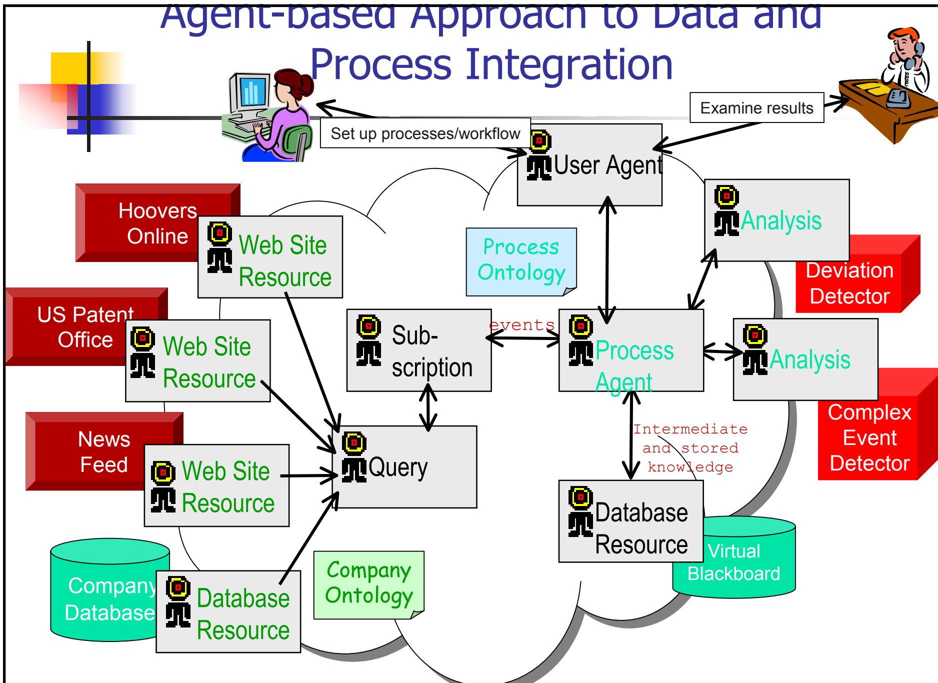
```
<advertisement>
  <capability NAME-"ServiceAgent5Cap">
    <ontology_fragment NAME="_infoSleuth" VERSION="1.0"
      <class NAME="agent"
        <slot NAME="agent address" VALUE="tcp:research.telcordia.com:7000"</slot>
        <slot NAME="agent name" VALUE="ServiceAgent5"></slot>
        <slot NAME="type" VALUE="serviceagent"></slot></class>
    </ontology_fragment>
    <ontology_fragment NAME="_conversation" VERSION="1.0">
      <class NAME="conversation">
        <slot NAME="type">
          <set_constraint><![CDATA["ask-all", "ask-one", "subscribe"]]></set_constraint>
         <slot NAME="message" VALUE="SOAP"></slot</class></ontology_fragment>
    <ontology_fragment NAME="_trip_planning _services" VERSION="1.0">
      <class NAME="domain">
         <slot NAME="domainSynonym" VALUE="travel"></slot>
         <slot NAME="rootDomain" VALUE="tourism"></slot>/class>
      <class NAME="booking-flight-ticket"></class>
         <slot NAME="operation" VALUE="Find-Ticket"></slot>
         <slot NAME="INPARAM1" VALUE="DepartingAirport" TYPE="String"></slot>
              <constraint><states><value>Texas</value>
                                   <value>California</value>  </states></constraints>
         <slot NAME="INPARAM2" VALUE="ArrivalAirport" TYPE="String"></slot>
         <slot NAME="OUTPARMA1" VALUE="Preice" TYPE="float"></slot>
          <slot NAME="operation" VALUE="Book-Ticket"></slot>
                ……
       </class>
    </ontology_fragment>
```

WebService Agents need to support Conversational Interactions

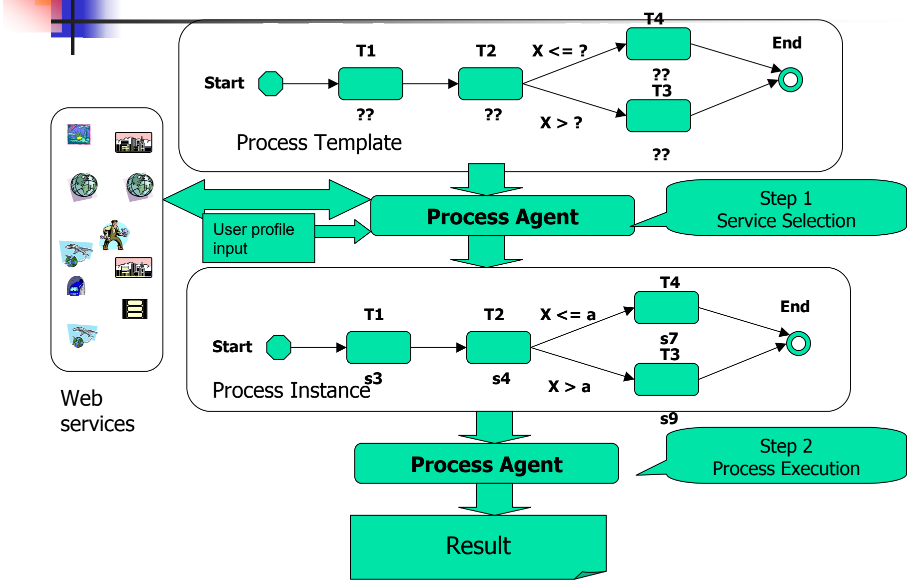


- Service interface captures abstractions of external applications using ***state machine***
- The service proxy is an ***adapter*** to external systems
- States and operations defined in the interface are used to control and ***converse*** with the external service

Agent-based Approach to Data and Process Integration



Process Agent



Dynamic generation of composite Web Service (workflow)

- One of the fundamental assumptions in WFMS is that workflow schema or process must be predefined.
- It is a daunting task to predefined every possible workflows with every possible possibilities.
- Due to frequent changing business conditions, it is necessary to alter or modify business processes on the fly.
- This implies the need for dynamic generation of workflow.
- One approach is to define business rules and business objective and generate workflow dynamically by using backward chain rules, forward chain rules, service selection rules and data flow rules.

Other Research Projects

- Process-based approach to semantic B2B Integration
- Peer-to-peer provisioning of dynamic web services
- Multimedia databases
  - modeling and querying of moving objects
  - indexing scheme to support fast and accurate retrieval of multimedia data