2003

Semantic Web Processes

Jorge Cardoso

Amit P. Sheth
Wright State University - Main Campus, amit@sc.edu

Follow this and additional works at: https://corescholar.libraries.wright.edu/knoesis

Part of the Bioinformatics Commons, Communication Technology and New Media Commons, Databases and Information Systems Commons, OS and Networks Commons, and the Science and Technology Studies Commons

Repository Citation

This Presentation is brought to you for free and open access by the The Ohio Center of Excellence in Knowledge-Enabled Computing (Kno.e.sis) at CORE Scholar. It has been accepted for inclusion in Kno.e.sis Publications by an authorized administrator of CORE Scholar. For more information, please contact library-corescholar@wright.edu.
Semantic Web Processes

Jorge Cardoso¹, Amit Sheth²,³
¹University of Madeira
²LSDIS Lab, Computer Science, University of Georgia
³Semagix, Inc

Net Object Days 2003 (NODe 2003)
September 22 to 25, 2003
Thuringia, Germany
Our Focus (1)

- Web services and their composition into Web Processes promise to power eCommerce and eServices

- Supporting Web Processes on multi-enterprise and Web scale require addressing heterogeneity/integration, scalability, dynamic change and performance challenges

- Semantics is seen as the key enabler to address these challenges; Semantic Web Processes build upon Web Services and Semantic Web technologies

- This tutorial is about adding *semantics* to *Web Services*, and exploiting them in *Web Process Lifecycle (Specification, Discovery, Composition, Execution)*

  - Functional perspective takes form of process composition involving *Web Service Discovery*, addressing semantic heterogeneity handling

  - Operational perspective takes form of the research on *QoS Specification* for Web Services and Processes.
Our Focus (2)

Semantics

Web Processes
- Web Process Composition
- Web Process QoS

Web Services
- Web Service Annotation
- Web Service Discovery
- Web Service QoS
What are Web Services, Web Processes, and Semantics?
“Web services are a new breed of Web application. They are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. Web services perform functions, which can be anything from simple requests to complicated business processes. …

Once a Web service is deployed, other applications (and other Web services) can discover and invoke the deployed service.”

*IBM web service tutorial*
Characteristics of Web-Services

- **Modular**: Components are reusable and it is possible to compose them into larger components.
- **Available**: Services must be exposed outside of the particular paradigm or system they are available in. Business services can be completely decentralized and distributed over the Internet. The dynamic enterprise and dynamic value chains become achievable.
- **Described**: Services have a machine-readable description that can be used to identify the interface of the service.
- **Implementation-independent**: The service interface is independent of the ultimate implementation.
- **Published**: Service descriptions are made available in a repository where users can find the service and use the description to access the service.

Fremantle et al. 2002, Enterprise Services, CACM. Oct
Why Web Services?

Web Services

UDDI

Web services

WSDL

SOAP

Jini

RMI (Remote Method Invocation)

Enterprise Java Beans

Microsoft DCOM

CORBA (Common Object Request Broker Architecture)

Open Software Foundation DCE (Distributed Computing Environment)

Sun ONC/RPC (Open Network Computing)

IP, UDP, TCP
### Why Web services?

<table>
<thead>
<tr>
<th>Feature</th>
<th>CORBA</th>
<th>Web Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Model</td>
<td>Object Model</td>
<td>SOAP Message exchange model</td>
</tr>
<tr>
<td>Client Server Coupling</td>
<td>Tight Coupling</td>
<td>Loose Coupling</td>
</tr>
<tr>
<td>Parameter Passing</td>
<td>Pass by reference/value</td>
<td>Pass by value only</td>
</tr>
<tr>
<td>Type Checking</td>
<td>1. Static + Runtime type checking (Regular) 2. Runtime type checking only (DII)</td>
<td>RunTime type checking only</td>
</tr>
<tr>
<td>State</td>
<td>Stateful</td>
<td>1. Stateless, Uncorrelated (Web Services) 2. Stateful (Web Process)</td>
</tr>
<tr>
<td>Firewall Traversal</td>
<td>Work in Progress</td>
<td>Uses HTTP port 80</td>
</tr>
<tr>
<td>Service Discovery</td>
<td>CORBA naming/trading Service</td>
<td>UDDI</td>
</tr>
<tr>
<td>Communication Mode</td>
<td>1-way, 2-way sync 2-way async</td>
<td>2-way sync (Web Services) 1-way, 2-way sync, 2-way async (Web Process)</td>
</tr>
</tbody>
</table>

Gokhale et al, Reinventing the Wheel ? CORBA vs Web-services; Sheth and Miller, [Web Services: Incremental Technical Advance with Huge Practical Impact](#)
What are Web Processes (1)?

- **Web Processes** are next generation workflow technology to facilitate the interaction of organizations with markets, competitors, suppliers, customers etc. supporting enterprise-level and core business activities
  - encompass the ideas of both intra and inter organizational workflow.
  - created from the composition of Web services

- When all the tasks involved in a Web process are semantically described, we may call such process as **Semantic Web Processes**
Web processes describe how Web services are connected to create reliable and dependable business solutions.

Web processes allow businesses to describe sophisticated processes that can both consume and provide Web services.

The role of Web processes within the enterprise is to simplify the integration of business and application processes across technological and corporate domains.
Web Process
An Example

- Graphical example of a web process

The BarnesBookPurchase process
Web Process
Another Example

Web Processes

Organization A

$t_1$  
Setup

$t_2$  
Prepare Sample

$t_3$  
Prepare Clones and Sequence

$t_4$  
Assembly

$t_5$  
Test Quality

$t_6$  
Get Sequences

Organization B

$t_7$  
Sequence Processing

$t_8$  
Process Report

Organization C
Web Processes Composition

Web Processes

Web Process Design

Web services
Globalization of Processes

Workflows

Distributed Workflows

B2B

E-Services

Enterprise

Inter-Enterprise

Web Processes

Global

Processes driving the Networked Economy
Stages of architectural evolution

- **Process Portal**
  - One stop for e-services, p2p interactions between buyer and sellers
  - E-Gov, industry automation, Life Science

- **Process Vortex**
  - Interactions between buyer and seller through a third party marketmaker, predefined processes, shared ontology

- **Dynamically Trading Processes**
BIG 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 → Web)
  - **Needs**: Automated service discovery/selection and composition

**Proposition**: Semantics is the most important enabler to address these challenges
What are Semantics and Ontologies?

An ontology includes a **vocabulary of terms**, and some **specification of their meaning**.

The goal is to create an **agreed-upon vocabulary** and semantic structure for exchanging information about that domain.
Roadmap

Semantic Web

Web Process Composition

Annotation of Web Services

Web Service Discovery

Web Processes Quality of Service
Semantics for Web Processes

- **Data/Information Semantics**
  - **What:** Formal definition of data in input and output messages of a web service
  - **Why:** for discovery and interoperability
  - **How:** by annotating input/output data of web services using ontologies

- **Functional/Operational Semantics**
  - Formally representing capabilities of web service
  - for discovery and composition of Web Services
  - by annotating operations of Web Services as well as provide preconditions and effects; Annotating TPA/SLA (future work)

- **Execution Semantics**
  - Formally representing the execution or flow of services in a process or operations in a service
  - for analysis (verification), validation (simulation) and execution (exception handling) of the process models
  - using State Machines, Petri nets, activity diagrams etc.

- **QoS Semantics**
  - Formally describing operational metrics of a web service/process
  - To select the most suitable service to carry out an activity in a process
  - using QoS model [Cardoso and Sheth, 2002] for web services
Semantics for Web Process Life-Cycle

Data / Information Semantics

Development / Description / Annotation
- WSDL, WSEL
- DAML-S
- Meteor-S (WSDL Annotation)

Composition (Choreography?)
- BPEL, BPML, WSCI, WSCL, DAML-S, METEOR-S (SCET, SPTB)

Execution (Orchestration?)
- BPWS4J, Commercial BPEL Execution Engines, Intalio n3, HP eFlow

Publication / Discovery
- UDDI
- WSIL, DAML-S
- METEOR-S (P2P model of registries)
Semantics for Web Process Life-Cycle

- **Execution (Orchestration?)**
  - BPWS4J, Commercial BPEL Execution Engines, Intalio n3, HP eFlow

- **Composition (Choreography?)**
  - BPEL, BPML, WSCI, WSCL, DAML-S, METEOR-S (SCET, SPTB)

- **Data / Information Semantics**

- **Development / Description / Annotation**
  - WSDL, WSEL, DAML-S, Meteor-S (WSDL Annotation)

- **Publication / Discovery**
  - UDDI, WSIL, DAML-S, METEOR-S (P2P model of registries)
Functional / Operational Semantics

Composition (Choreography?)
BPEL, BPML, WSCI, WSCL, DAML-S, METEOR-S (SCET,SPTB)

Execution (Orchestration?)
BPWS4J, Commercial BPEL Execution Engines, Intalio n3, HP eFlow

Development / Description / Annotation
WSDL, WSEL, DAML-S, Meteor-S (WSDL Annotation)

Publication / Discovery
UDDI, WSIL, DAML-S, METEOR-S (P2P model of registries)
Semantics for Web Process Life-Cycle

**Execution (Orchestration?)**
BPWS4J, Commercial BPEL Execution Engines, Intalio n3, HP eFlow

**Development / Description / Annotation**
WSDL, WSEL DAML-S Meteor-S (WSDL Annotation)

**Composition (Choreography?)**
BPEL, BPML, WSCI, WSCL, DAML-S, METEOR-S (SCET,SPTB)

**QoS Semantics**

**Publication / Discovery**
UDDI WSIL, DAML-S METEOR-S (P2P model of registries)
Semantics for Web Process Life-Cycle

Execution (Orchestration?)

Composition (Choreography?)

Execution Semantics

Development / Description / Annotation

Publication / Discovery

BPWS4J, Commercial BPEL Execution Engines, Intalio n3, HP eFlow

BPEL, BPML, WSCI, WSCL, DAML-S, METEOR-S (SCET,SPTB)

WSDL, WSEL DAML-S
Meteor-S (WSDL Annotation)

UDDI
WSIL, DAML-S
METEOR-S (P2P model of registries)
Semantics for Web Process Life-Cycle

- **Execution (Orchestration?)**
  - BPWS4J, Commercial BPEL Execution Engines, Intalio n3, HP eFlow

- **Composition (Choreography?)**
  - BPEL, BPML, WSCI, WSCL, DAML-S, METEOR-S (SCET, SPTB)

- **Development / Description / Annotation**
  - WSDL, WSEL DAML-S Meteor-S (WSDL Annotation)

- **Publication / Discovery**
  - UDDI WSIL, DAML-S METEOR-S (P2P model of registries)

<table>
<thead>
<tr>
<th>Execution Semantics</th>
<th>Data / Information Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>QoS Semantics</td>
<td>Functional / Operational Semantics</td>
</tr>
</tbody>
</table>

Semantics Required for Web Processes

- Development
- Description
- Annotation

- Execution
- Composition
- Publication / Discovery

- Semantics
  - Functional / Operational Semantics
  - Data / Information Semantics
  - QoS Semantics
  - Execution Semantics

- Semantics for Web Process Life-Cycle
Web Processes Architecture

How can semantics be exploited???

Web Servers
HTTP/HTTPS
TCP/IP-SSL

Semantics

WSDL
SOAP
XML

... 

Composition

Brokering
Discovery
UDDI
Semantics for Data Resources

Semantic Web servers

Utilize ontology supported annotation of web resources

Semantic Web browsers

Making sense of page contents
Supporting the interpretation of web pages

Web Servers
HTTP/HTTPS
TCP/IP-SSL

Semantics

WS9
Web page

WS2
Web page
Web Process Architecture

Web Service Semantic Annotation

Associate ontological concepts to Web Service descriptions

METEOR-S

Semantics

SOAP

Semantics

WSDL

XML

DAML-S

Adding Semantics to Web Services Standards, Semantic Annotation of Web Services
Web Services

- **WSDL** defines services as collections of network endpoints or *ports*. A port is defined by associating a network address with a binding; a collection of ports define a service.

- **SOAP** is a message layout specification that defines a uniform way of passing XML-encoded data. It also defines a way to bind to HTTP as the underlying communication protocol. SOAP is basically a technology to allow for “RPC over the web”.

- **XML** was designed to represent data.
- **WSDL** stands for Web Services Definition Language
- **WSDL** is an XML document
- **WSDL** is used to describe Web services
- **WSDL** is also used to access Web services
<definitions>
  <types>
    definition of types..
  </types>
  <message>
    definition of messages...
  </message>
  <portType>
    <operation> ..... </operation>
    <operation> ..... </operation>
  </portType>
  <binding>
    definition of binding....
  </binding>
  <service>
    <port>....</port>
    <port>....</port>
  </service>
</definitions>

From S. Chandrasekaran's Talk
To enhance the discovery, composition, and orchestration of Web services, it is necessary to provide richer description of their interfaces.

One solution is to annotate WSDL interfaces with semantic metadata based on/using relevant ontologies.

An ontology is a specification of a representational vocabulary for a shared domain of discourse.
Description Layer:

Why:
- Unambiguously understand the functionality of the services and the semantics of the operational data

How:
- Using Ontologies to semantically annotate WSDL constructs (conforming to extensibility allowed in WSDL specification version 1.2) to sufficiently explicate the semantics of the
  - data types used in the service description and
  - functionality of the service

Present scenario:
- WSDL descriptions are mainly syntactic (provides operational information and not functional information)
- Semantic matchmaking is not possible
How to Annotate?

- Map Web service’s input & output data as well as functional description using relevant data and function/operation ontologies, respectively.

- How?
  - Borrow from schema matching
  - Semantic disambiguation between terms in XML messages represented in WSDL and concepts in ontology.
A Web service (WS) invocation specifies:
- The number of input parameters that must be supplied for a proper WS realization and
- The number of outputs parameters to hold and transfer the results of the WS realization to other tasks.
- A function to invoke

```
function_foo(x..y)
```
Types of Annotation

Functional Semantics

- Client
- Local
- Tourism

Data Semantics

- Receipt
- Itinerary

QoS Semantics

- Security
- Time
- Duration
- Price
- Cost
- Fidelity
- Repudiation
- Availability
- Reliability

function_foo(x..y)
Adding Semantics to Web Services

**WSDL**

```xml
<xsd:complexType name="Date">
  <xsd:sequence>
    <xsd:element name="year" type="xsd:integer" />
    <xsd:element name="month" type="xsd:integer" />
    <xsd:element name="day" type="xsd:byte" />
  </xsd:sequence>
</xsd:complexType>
```

**Ontologies**

- **Temporal-Entity**
  - **Time-Point** (`{absolute_time}`)
  - **Time** (`{hour, minute, second}`)
- **Scientific-Event** (`{millisecond}`)
- **Calendar-Date** (`{dayOftheWeek, monthOftheYear}`)

**Web Service**

- Inputs: Name, Year, Duration, City
- Outputs: Date, Interfaces

**QoS Ontology**

- **Get Conference Information**

**Data Semantics**

- XML Schema
- Data type hierarchy

**Functional Semantics**

- WSDL
- Quality

**QoS Semantics**

- Conference Information Functions
- Min
SOAP is an XML Messaging Protocol that allows software running on disparate operating systems, running in different environments to make procedure calls.

```
<SOAP:Envelope
 xmlns:SOAP='http://schemas.xmlsoap.org/soap/envelope/'
 SOAP:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'
 xmlns:v='http://www.topxml.com/soapworkshop/'>
  <SOAP:Header>
    <v:From SOAP:mustUnderstand='1'>
      cdix@soapworkshop.com
    </v:From>
  </SOAP:Header>
  <SOAP:Body>
    <v:DoCreditCheck>
      <ssn>123-456-7890</ssn>
    </v:DoCreditCheck>
  </SOAP:Body>
</SOAP:Envelope>
```
Why SOAP?

- Today's applications communicate using Remote Procedure Calls (RPC) between objects like DCOM and CORBA.

- RPC represents a compatibility and security problem; firewalls and proxy servers will normally block this kind of traffic.

- A better way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. SOAP was created to accomplish this.
<soap:Body>
  <m:GetPrice xmlns:m="http://www.w3schools.com/prices">
    <m:Item>Apples</m:Item>
  </m:GetPrice>
</soap:Body>

Data Semantics
Web Process Architecture

Semantic Brokering

Specialized brokering services to find Web services

Semantic Discovery

Discovery algorithms that account for semantic information

Semantic Registries

Describe Web services in UDDI registries using semantic concepts

METEOR-S

Semantics

Brokering

Semantics

Discovery

Semantics

UDDI

Semantics
- **UDDI** stands for Universal Description, Discovery and Integration

- **UDDI** serves as a “Business and services” registry and directory and are essential for dynamic usage of Web services

- A **UDDI** registry is similar to a CORBA trader, or it can be thought of as a DNS for business applications.

- Is a platform-independent framework for describing services, discovering businesses, and integrating business services by using the Internet.
How UDDI Works?

1. SW companies, standards bodies, and programmers populate the registry with descriptions of different types of services.

2. Businesses populate the registry with descriptions of the services they support.

3. UBR assigns a programmatically unique identifier to each service and business registration.

4. Marketplaces, search engines, and business apps query the registry to discover services at other companies.

5. Business uses this data to facilitate easier integration with each other over the Web.

Publication and Discovery Layers:

Why:
- Enable scalable, efficient and dynamic publication and discovery (machine processable / automation)

How:
- Use of ontology to categorize registries based on domains and characterize them by maintaining the
  1. properties of each registry
  2. relationships between the registries
- Capturing the WSDL annotations in UDDI

Present scenario:
- Suitable for simple searches (like services offered by a provider, services that implement an interface, services that have a common technical fingerprint etc.)
- Categories are too broad
- Automated service discovery (based on functionality) and selecting the best suited service is not possible
UDDI and Semantics

Marketplaces, search engines, and business apps query

Semantic UDDI

Registry entry

Functional Semantics

Data Semantics

QoS Semantics

Internet

WS\textsubscript{9}

WS\textsubscript{4}

WS\textsubscript{8}

function\_foo(x..y)

Inputs

QoS

Outputs

Client

Receipt

Local

Itinerary

Tourism

Security

Price

Time

Cost

Fidelity

Reliability

Repudiation

Availability
Web Services must be located (Discovery) that might contain the desired functionality, operational metrics, and interfaces needed to carry out the realization of a given task.
Discovery
New Requirements
State of the art in discovery

UDDI Business Registry

Provides non-semantic search

Search

Keyword and attribute-based match

Selection

Which service to select? How to select?

Results

Search retrieves lot of services (irrelevant results included)
Present Discovery Mechanism

Keyword and attribute-based search

- UDDI: Keyword and attribute-based search
- Example: “Quote”
  - Microsoft UBR returned 12 services
  - Human reading of description (Natural Language) help me understand:
    - 6 Entries are to get Famous Quotes
    - 1 Entry for personal auto and homeowners quoting
    - 1 Entry for multiple supplier quotes on all building materials

- Categorization suggested for UDDI is useful but inadequate (what does the WS do?):
  - 1 Entry for Automobile Manufacturing
  - 1 Entry for Insurance agents, brokers, & service

- Alternatively read and try to understand WSDL
  - 1 Entry related to security details (Human Understanding)
  - 1 Test Web service for Quotes (which quote?)
Present Discovery Mechanism
Search for services to book an air ticket (using categories)*

- unspsc-org: unspsc:3-1
  - Travel, Food, Lodging and Entertainment Services
    - Travel facilitation
      - Travel agents
        - Travel agencies

- Services: 3 records found.
  - AirFares
    Returns air fares from netviagens.com travel agent
  - Hotel reservations
    Reservations for hotels in Asia, Australia and New Zealand
  - Your Vacation Specialists
    Web enabled vacation information

- Providers: 2 records found.

* Search carried out in one of the Universal Business Registries
Present Discovery Mechanism
Search for services to book an air ticket (using Keywords)*

- air ticket
  - 1 record with name air tickets booking (service name happened to match keyword)

- airticket, ticketbooking, airtravel, air travel, travel agent, airticketbooking, air ticket booking, travel agency, travelagency
  - 0 records were returned

- travelagent
  - 1 record with name travelagent test
    - 4 services: BookFlight, cancelFlightBooking etc.
    - Descriptions say that both these services are “XML based Web services”
    - No URL for WSDL

- Travel
  - 15 records. Purpose/functionality understood from descriptions
    - 2 services : TravelBooks
    - 4 services : TravellInformation
    - 2 services : Reservation and cancellation of travel tickets
    - 1 service : Emergency Services for travellers
    - 1 service : Travel documentation and itinerary
    - 5 services : Description is ambiguous/not present

* Search carried out in one of the Universal Business Registries
The use of semantics

Benefits

- Search engines can better “understand” the contents of a particular service.
- More accurate searches.
- Additional information aids precision.
- Makes it possible to automate searches because less manual “weeding” is needed to process the search results.
- Facilitates the integration of several Web services.
Semantic Discovery: Overview

- **Annotation and Publication**
  - WSDL file is **annotated** using ontologies and the annotations are captured in UDDI

- **Discovery**
  - Requirements are captured as **templates** that are constructed using ontologies and semantic matching is done against UDDI entries
    - Functionality of the template, its inputs, outputs, preconditions and effects are annotated using ontologies

- **Use of ontologies**
  - Brings service provider and service requestor to a **common conceptual space**
  - Helps in **semantic matching** of requirements and specifications
Use of ontologies enables shared understanding between the service provider and service requestor.

For simplicity of depicting, the ontology is shown with classes for both operation and data.
Discovery in Semantic Web
Using Semantics

- **Functionality**: What capabilities the distributor expects from the service (Functional semantics)
- **Inputs**: What the distributor can give to the Manufacturer’s service (Data semantics)
- **Outputs**: What the distributor expects as outputs from the service (Data semantics)
- **QoS**: Quality of Service the distributor expects from the service (QoS semantics)

- **Description**: Natural language description of the service functionality (Syntactic description)
Syntactic, QoS, and Semantic (Functional & Data) Similarity

Syntactic Similarity

\[ \text{Syntactic Similarity} = \frac{\text{sn} + \omega_2 \cdot \text{SynDS}(\text{ST}, \text{sd}, \text{SO}, \text{sd})}{\omega_1 + \omega_2} \in [0..1], \]
and \( \omega_1, \omega_2 \in [0..1] \)

QoS Similarity

\[ \text{QoS Similarity} = \text{ST} \cdot \text{sd} \cdot \text{SO} \cdot \text{cost} \times \text{QoSdimD} \]

Functional & Data Similarity

Web Service

Web Service

Web Service

Web Service

Web Service

Web Service

Web Service

Web Service

Web Service
Semantic Discovery

- **IOParametersMatch** \((w,o) = \)
  - LinguisticMatch \((w,o) + StructureMatch (w,o) + ContextMatch (w,o)\)

- **LinguisticMatch** \((w,o) => \)
  - NameMatch
  - SynonymsMatch
  - HypernymRelation \((w \text{ is a kind of } o)\)
  - HyponymRelation \((o \text{ is a kind of } w)\)

- **StructureMatch** \((w,o) => \)
  - subTree\((w) == \text{subTree}(o)\)

- **ContextMatch**
  - Name of the parent concept provides some insight to the context of the term
IOParametersMatch \( (w,o) = \)
\[
  w_1 \cdot \text{LinguisticMatch} (w,o) + w_2 \cdot \text{StructureMatch} (w,o) + w_3 \cdot \text{ContextMatch} (w,o)
\]
\[
  w_1 + w_2 + w_3
\]

- Weights \( w_1, w_2 \) and \( w_3 \) can be decided by the user based on the confidence in the respective type of matching technique.
- LinguisticMatch uses a SynonymDictionary, also uses WordNet.

StructureMatch \((w, o) = \)

\[
\begin{align*}
  \text{max} &< \text{LinguisticMatch}(w,o) \quad \text{if} \quad o \in O.\text{subclasses} \\
  \sqrt{\text{LinguisticMatch}(w,o) \cdot \text{RangeMatch}(w,o)} &\quad \text{if} \quad o \in O.\text{properties}
\end{align*}
\]
The key players of brokering are the service providers, service consumers, and facilitators.

- **Providers** advertise their web services.
- **Facilitators** matches subscriptions to advertised services.
- **Consumers** register web services needs.
Semantic Brokering Issues
(for any resource: data or service)

- Structured and non-structured sources
- Transparency
  - Location, schema, language, and ontologies
- Query models
  - Semantic-based, rule-based, SQL-like, etc
- Semantic Mediators
  - Semantic query analysis and query processing
  - Use wrappers
Brokering and Semantics

- Find Web services across several UDDIs
- Specialized and optimized brokers for specific domain search
  - Transports, Finances, Education, etc.
- Allow the interpretation of complex requirements
  - Domain semantics
  - Functional semantics
  - Data semantics
  - QoS semantics
Semantic Composition

Use several Web services to achieve broader objective; requires algorithms to match Web service compatibility and perform integration.

METEOR-S

Composition

Semantics
Composition is the task of combining and linking existing Web Services and other components to create new processes.

Types of Composition

- **Static Composition** - services to be composed are decided at design time
- **Dynamic Composition** - services to be composed are decided at run-time

SCET, Semantic Web Process Composition
Composition of Web Processes

Once the desired Web Services have been found (Discovery), mechanisms are needed to facilitate the resolution of structural and semantic differences (integration).

This is because the heterogeneous Web services found in the first step need to interoperate with other components present in a process.
Flow Layer:

**Why:**
- Design (composition), **analysis** (verification), **validation** (simulation) and **execution** (exception handling) of the process models
- To employ mediator architectures for automated composition, control flow and data flow based on requirements
- To employ user interface to capture template requirements and generate template based on that

**How:**
- Using
  - Functionality/preconditions/effects of the participating services
  - Knowledge of conversation patterns supported by the service
  - Formal mathematical models like process algebra, concurrency formalisms like State Machines, Petri nets etc.
  - **Simulation** techniques

**Present Scenario:**
- Composition of Web services is static.
- Dynamic service discovery, run-time binding, analysis and simulation are not supported directly
Integration Requirements

- When Web services are put together
  - Their interfaces need to interoperate.
  - Structural and semantic heterogeneity need to be resolved*.

- **Structural heterogeneity** exists because Web services use different data structures and class hierarchies to define the parameters of their interfaces.

- **Semantic heterogeneity** considers the intended meaning of the terms employed in labeling interface parameters. The data that is interchanged among Web services has to be understood.

* Kashyap and Sheth 1996
How to establish data connections between Web Services interfaces?

How to establish data connections between the different data structures and class hierarchies of the interface parameters?

How to understand the intended meaning of the terms used in labeling interface parameters?
To enhance the integration, Web services need to have their inputs and outputs associated with ontological concepts (annotation).

This will facilitate the resolution of structural and semantic heterogeneities.

Compute the optimal matching (Bondy and Murty, 1976) using semantic information (Cardoso and Sheth, 2002).

Bipartite graph. Each edge has a weight (semantic similarity).
Semantic Web Processes

Questions?
Coffee Break
10 Minutes

NEXT: Composition Languages
NEXT: METEOR-S
Composition Languages

- BPEL4WS
- DAML-S
BPEL4WS (Business Process Execution Language for Web Services) is a process modeling language.
- Developed by IBM, Microsoft, and BEA
- Version 1.1, 5 May 2003

It supercedes XLANG (Microsoft) and WSFL(IBM).

It is build on top of WSDL.
- For descriptions of what services do and how they work, BPEL4WS references port types contained in WSDL documents.
• DAML-S The service profile ontology describes the functionality of a Web service.

* Fensel and Bussler 2002
BPEL4WS

Introduction

- BPEL4WS was released along with two others specs:
  - WS-Coordination and WS-Transaction*.

- **WS-Coordination** describes how services can make use of pre-defined coordination contexts to subscribe to a particular role in a collaborative activity.

- **WS-Transaction** provides a framework for incorporating transactional semantics into coordinated activities.

BPEL4WS

Introduction

- BPEL4WS is a **block-structured programming language**, allowing recursive blocks but restricting definitions and declarations to the top level.

- The language defines **activities** as the basic components of a process definition.

- Structured activities prescribe the order in which a collection of activities take place.
  - Ordinary sequential control between activities is provided by **sequence**, **switch**, and **while**.
  - Concurrency and synchronization between activities is provided by **flow**.
  - Nondeterministic choice based on external events is provided by **pick**.
- Process instance-relevant data (containers) can be referred to in routing logic and expressions.

- BPEL4WS defines a mechanism for catching and handling faults similar to common programming languages, like Java.

- One may also define a compensation handler to enable compensatory activities in the event of actions that cannot be explicitly undone.

- BPEL4WS does not support nested process definition.
Let consider the following process.

<definitions targetNamespace="http://manufacturing.org/wsd1/purchase"
    xmlns:ns="http://manufacturing.org/xsd/purchase"
...
<message name="POMessage">
    <part name="customerInfo" type="ns:customerInfo"/>
    <part name="purchaseOrder" type="ns:purchaseOrder"/>
</message>
...
<message name="scheduleMessage">
    <part name="schedule" type="ns:scheduleInfo"/>
</message>

<portType name="purchaseOrderPT">
    <operation name="sendPurchaseOrder">
        <input message="pos:POMessage"/>
        <output message="pos:InvMessage"/>
        <fault name="cannotCompleteOrder"
            message="pos:orderFaultType"/>
    </operation>
</portType>
...
<slnk:serviceLinkType name="purchaseLT">
    <slnk:role name="purchaseService">
        <slnk:portType name="pos:purchaseOrderPT"/>
    </slnk:role>
</slnk:serviceLinkType>
...
</definitions>

The WSDL portType offered by the service to its customer

Messages

Roles
<process name="purchaseOrderProcess"
    targetNamespace="http://acme.com/ws-bp/purchase"
...>
    <partners>
        <partner name="customer"
            serviceLinkType="lns:purchaseLT"
            myRole="purchaseService"/>
    </partners>
</process>

This section defines the different parties that interact with the business process in the course of processing the order.

This section defines the data containers used by the process, providing their definitions in terms of WSDL message types.

This section contains fault handlers defining the activities that must be executed in response to faults.
...<sequence>

<receive partner="customer"
    portType="lns:purchaseOrderPT"
    operation="sendPurchaseOrder"
    container="PO">
</receive>

<flow>
...
</flow>

<reply partner="customer"
    portType="lns:purchaseOrderPT"
    operation="sendPurchaseOrder"
    container="Invoice"/>

</sequence>

</process>
<flow>
  <links>
    <link name="ship-to-invoice"/>
    <link name="ship-to-scheduling"/>
  </links>
  <sequence>
    ...
    <invoke partner="shippingProvider"
             portType="lns:shippingPT"
             operation="requestShipping"
             inputContainer="shippingRequest"
             outputContainer="shippingInfo">
      <source linkName="ship-to-invoice"/>
    </invoke>
    <receive partner="shippingProvider"
             portType="lns:shippingCallbackPT"
             operation="sendSchedule"
             container="shippingSchedule">
      <source linkName="ship-to-scheduling"/>
    </receive>
    ...
  </sequence>
</flow>
DAML-S

Introduction

- DAML-S
  - DAML (DARPA Agent Markup Language)
  - DAML-S: Upper ontology of web services

- DAML-S provides support for the following elements:
  - Process description.
  - Advertisement and discovery of services.
  - Selection, composition & interoperation.
  - Invocation.
  - Execution and monitoring.

DAML-S project home page
- DAML-S defines ontologies for the construction of service models:
  - Service Profiles
  - Process Models
  - Service Grounding

Diagram:

- Resource
  - Service
    - ServiceProfile
      - what the service does
    - ServiceModel
      - how the service works
    - Service Grounding
      - how to access the service

Arrow labels:
- provides
- presents
- described by
- supports
The Service Profile provides details about a service.

**Inputs.** Inputs that should be provided to invoke the service.

**Outputs.** Outputs expected after the interaction with the service.

**Preconditions.** Set of conditions that should hold prior to the service being invoked.

**Effects.** Set of statements that should hold true if the service is invoked successfully.
An example of Inputs and Outputs

...
BPEL4WS vs. DAML-S

Comparison

- BPEL4WS relates closely to the ServiceModel (Process Model) component of DAML-S.

- DAML-S defines preconditions and effects
  - This enables the representation of side effects of Web services.
  - It also enables a better reasoning about the composition of services.

- DAML-S classes provide a richer representation of services
  - Classes allow reasoning draw properties from inheritance and other relationships to other DAML-S classes.
BPEL4WS vs. DAML-S

Comparison

- The DAML-S ServiceProfile and ServiceModel provide sufficient information to enable
  - The automated discovery, composition, and execution based on well-defined descriptions of a service's inputs, outputs, preconditions, effects, and process model.

- BPEL4WS has *complicated semantics* for determining whether an activity actually happens in a block.

- BPEL4WS defines mechanisms for *catching and handling faults* and for setting compensation handlers.

- BPEL4WS includes *WS-Coordination* and *WS-Transaction* to provide a context for pre-defined transactional semantics.
Organizations operating in modern markets, such as e-commerce activities, require QoS management.

QoS management is indispensable for organizations striving to achieve a higher degree of competitiveness.
Defining process operational metrics at design time is difficult. Most WS do not specify them, autonomy and heterogeneity makes matter worse.

Nevertheless, when composing a process it is indispensable to inquire the Web services operational metrics.

Operational metrics characterize the Quality of Service (QoS) that Web services exhibit when invoked.
QoS
New Requirements

Before

Time: 17 Hours
Cost?
Reliability?
Fidelity?

Now

Time?
Cost?
Reliability?
Fidelity?
QoS Semantics

- **What?**
  Formally describes operational metrics of a web service/process

- **Why?**
  To select the most suitable service to carry out an activity in a process

- **How?**
  Using QoS model for web services

[Cardoso and Sheth, 2002]
QoS

Benefits

- **Composition** of processes according to QoS objective and requirements.

- **Selection** of services and execution of processes based on QoS metrics.

- **Monitoring** of processes to assure compliance with initial QoS requirements.

- **Evaluation** of alternative strategies when QoS requirements are violated.
Semantic WP QoS
Research Issues

**Specification.** What dimensions need to be part of the QoS model for processes?

**Computation.** What methods and algorithms can be used to compute, analyze, and predict QoS?

**Monitoring.** What kind of QoS monitoring tools need to be developed?

**Control.** What mechanisms need to be developed to control processes, in response to unsatisfactory QoS metrics?
Operational Metrics Specification

- Operational metrics are described using a QoS model represented with a suitable ontology.

The specification of Web services operational metrics allow the analysis and computation processes QoS.

Processes can be designed according to QoS objectives and requirements.

This allows organizations to translate their strategies into their processes more efficiently.
A QoS Model describes **non-functional** properties of a process

Which dimensions should be part of a QoS model?
QoS Models and Semantics

?  ?  ?  Z#$%&/  

Cost  Price  Security  Reliability  Time  Duration  Fidelity  Repudiation  Availability

Security  Time  Cost  Fidelity  Reliability  Price  Duration  Repudiation  Availability

Use Semantics

QoS
QoS in METEOR-S

QoS Model

QoS Estimates for Tasks/Web services

QoS Estimates for Transitions

Stochastic Process

Enact

Log

QoS Computation

SWR algorithm

Simulation

Design
To analyze a process QoS, it is necessary to:

- Create estimated for task QoS metrics and
- Create estimated for transition probabilities

Once tasks and transitions have their estimates set, algorithms and mechanisms, such as simulation, can be applied to compute the overall QoS of a process.
WS runtime behavior description can be composed of several classes. For example:

<table>
<thead>
<tr>
<th>QoS Model</th>
<th>Basic class</th>
<th>Distributional class</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Basic class</td>
<td>Distributional class</td>
</tr>
<tr>
<td>Time</td>
<td>0.291</td>
<td>Normal(0.674, 0.143)</td>
</tr>
<tr>
<td>Cost</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>Reliability</td>
<td>-</td>
<td>1.0</td>
</tr>
<tr>
<td>Fidelity.a_i</td>
<td>0.63</td>
<td>Trapezoidal(0.7,1,1,4)</td>
</tr>
<tr>
<td>Time</td>
<td>0.674</td>
<td>0.0</td>
</tr>
<tr>
<td>Cost</td>
<td>0</td>
<td>1.0</td>
</tr>
<tr>
<td>Reliability</td>
<td>100%</td>
<td>1.0</td>
</tr>
<tr>
<td>Fidelity.a_i</td>
<td>0.81</td>
<td>0.92</td>
</tr>
</tbody>
</table>

Task QoS for an automatic task (SP FASTA task)
Web process QoS computation

Design time| Runtime

- Linear programming
- Simulation
- Petri-nets analysis
- Graph Reduction Techniques
- Critical Path Algorithm
- Security
- Time
- Cost
- Fidelity
- Reliability
- Price
- Duration
- Repudiation
- Reliability
- Availability

QoS

Diagram showing the relationships between various techniques and QoS parameters.
QoS Computation

Graph Reduction Technique
Graph Reduction Technique

Reduction of a Sequential System

\[ T(t_{ij}) = T(t_i) + T(t_j) \]
\[ C(t_{ij}) = C(t_i) + C(t_j) \]
\[ R(t_{ij}) = R(t_i) \ast R(t_j) \]
\[ F(t_{ij}).a_r = f(F(t_i), F(t_j)) \]
QoS Computation

Graph Reduction Technique

**Reduction of a Parallel System**

\[ T(t_{ln}) = \text{Max}_{i \in \{1..n\}} \{T(t_i)\} \]

\[ C(t_{ln}) = \sum_{1 \leq i \leq n} C(t_i) \]

\[ R(t_{ln}) = \prod_{1 \leq i \leq n} R(t_i) \]

\[ F(t_{ln}).a_r = f(F(t_1), F(t_2), \ldots, F(t_n)) \]
● While mathematical methods can be effectively used, another alternative is to utilize simulation analysis\(^1\).

● Simulation can play an important role in tuning the QoS metrics of processes by exploring “what-if” questions.

● In our project, these capabilities involve a loosely-coupled integration between the METEOR WfMS and the JSIM simulation system\(^2\).

\(^1\)Miller, Cardoso et al. 2002, \(^2\)Nair, Miller et al. 1996; Miller, Nair et al. 1997; Miller, Seila et al. 2000.
Semantic Web Processes

Questions?

NEXT: METEOR-S Project @ LSDIS lab
Systems and Applications

METEOR-S Project @ LSDIS lab
Web Process Life-Cycle

1. Design
   - Create Process WSDL
   - Create Process Template and Add Activities
   - Find Ontologies & Annotate Activity Requirements
   - Add Control Flow

2. Discovery
   - Find Matches
   - Rank Services
   - Select a Service

3. Composition
   - Add to Process
   - Data Transformation
   - Data Flow

4. Execution
   - Generate Process
   - Validate Syntax
   - Execute

Steps:
1. Design
2. Discovery
3. Composition
4. Execution
METEOR-S exploits Workflow, Semantic Web, Web Services, and Simulation technologies to meet these challenges in a practical and standards based approach.

- Applying Semantics in Annotation, Quality of Service, Discovery, Composition, Execution of Web Services
- Adding semantics to different layers of Web services conceptual stack
- Use of ontologies to provide underpinning for information sharing and semantic interoperability

How to Annotate?

- Map Web service’s input/output data as well as functional description using relevant data and function/operation ontologies, respectively

- How?
  - Borrow from schema matching
  - Semantic disambiguation between terms in XML messages represented in WSDL and concepts in ontology
Semantic Annotation

- \( \text{IOParametersMatch} (w,o) = \) 
  \( \text{LinguisticMatch} (w,o) + \text{StructureMatch} (w,o) + \text{ContextMatch} (w,o) \)

- \( \text{LinguisticMatch} (w,o) \Rightarrow \)
  - NameMatch with stemming
  - Description Match
  - SynonymsMatch
  - HypernymRelation \((w \text{ is a kind of } o)\) : prevailing_speed is a type of speed of a wind i.e. windSpeed
  - HyponymRelation \((o \text{ is a kind of } w)\)
  - Acronyms : Sea Level Pressure has acronym SLP

- \( \text{StructureMatch} (w,o) \Rightarrow \)
  - \( \text{subTree}(w) == \text{subTree}(o) \)

- \( \text{ContextMatch} \)
  - Name of the parent concept provides some insight to the context of the term
Semantic Annotation: Data Semantics

```
<xsd:complexType name="Wind">
  <xsd:sequence>
    <xsd:element name="prevailing_speed" type="xsd:double" />
    <xsd:element name="gust_speed" type="xsd:double" />
    <xsd:element name="prevailing_direction" type="xsd1:Direction" />
  </xsd:sequence>
</xsd:complexType>

WindEvent

```

```
<xsd:complexType name="Pressure">
  <xsd:sequence>
    <xsd:element name="altimeter" type="xsd:double" />
    <xsd:element name="slp" type="xsd:double" />
    <xsd:element name="delta" type="xsd:double" />
  </xsd:sequence>
</xsd:complexType>

PressureEvent

```

```
WSDL : GlobalWeather.wsdl

Ontology : weather-ont.daml

Patil, Oundhakar, Sheth, SAWS Technical Report
```
Semantic Annotation: Functional Semantics

A Sample Functional Ontology

WeatherFunctions

- getWeather
  - getWind
  - getPressure
  - getTemperature

- getStation
  - getStationByCountry
  - getStationByZip

WSDL Operations

```xml
<portType name="GlobalWeather">
  <operation name="getWeatherReport">
    <input message="tns:getWeatherReport" />
    <output message="tns:getWeatherReportResponse" />
  </operation>
</portType>

<portType name="StationInfo">
  <operation name="searchByCountry">
    <input message="tns:searchByCountry" />
    <output message="tns:searchByCountryResponse" />
  </operation>
</portType>
```

Patil, Oundhakar, Sheth, SAWS Technical Report
METEOR-S components for Semantic Web Services

- **Discovery Infrastructure (MWSDI)**
  - Semantic Annotation and Discovery of Web Services \(^1\)
  - Semantic Peer-to-Peer network of Web Services Registries \(^2\)

- **Composer**
  - SCET: Service Composition and Execution Tool \(^3\)
  - Semantics Process Template Builder and Process Generator \(^4\)
  - QoS Management
    - Specify, compute, monitor and control QoS (SWR algorithm) \(^5\)

- **Orchestrator** (Under development)
  - Analysis and **Simulation** \(^6\)
  - Execution
  - **Monitoring** \(^6\)

---

\(^1\) [Sivashanmugam et al.-1], \(^2\) [Verma et al.], \(^3\) [Chandrasekaran et al.], \(^4\) [Sivashanmugam et al.-2], \(^5\) [Cardoso et al.], \(^6\) [Silver et al.]
- uses **Functional, Data** and **QoS** semantics

**Service Discovery**
Service Selection

- uses Functional, Data and QoS semantics
- needed for the world where business processes never stop changing
**Repositories are used to store**
1. Web Service Interfaces
2. Ontologies
3. Process Templates

**Process Execution**
1. Validation and deployment
2. Executing the process using a client

**Process Designer**
1. Template Construction
   activity specification using
   - interfaces
   - services
   - semantic activity templates
   - other details
2. Process Generation
   - Service discovery (automatic)
     and selection (semi-automatic)
   - Data flow

---

**Discovery Infrastructure**
(MWSDI)

**Repositories**
- Activity Interfaces
- Process Templates
- Ontologies

**Template Builder**

**Process Generator**

**Process Designer**

**Execution Engine**
SCET (Service Composition and Execution Tool) allows

- to compose services statically by modeling the process as a digraph in a graphical designer
- stores the process description as WSFL based specification
- allows execution of the composed process using Perl
- supports a simple execution monitoring feature
- supports performance estimation using JSIM simulation

Senthilanand Chandrasekaran, M.Sc. Thesis presented at the Department of Computer Science of the University of Georgia.
Simulation provides feedback on processes, allowing the composer to modify his process design by:

- **Replacing services** which do not satisfy the expected runtime behavior with more suitable Web services.
- **Modifying the process structure** (control flow) based on the simulation runs.

Senthilnand Chandrasekaran, M.Sc. Thesis presented at the Department of Computer Science of the University of Georgia.
Semantic Web Process Design

Template Construction
Semantic Web Process Design

Process Generation

[Image of a software interface with several tabs and fields for process design, including activity name, decomposable status, ontology URL, operation concept, discovery URL, and various specification files with buttons to open them.]

Add Message
Add Precondition
Update
Add Effect
Show Services
### Semantic Web Process Design

**Activity Name:** TertiarySupplierPartner

**Decomposable:**

**Ontology URL:** `http://localhost/~kaarthik/LSDIS-FunctionalOnt.daml`

**Operation Concept:** eForOrderToyParts

**Discovery URL:** `http://localhost/server/RegistryServerServlet`

**Discovery Specifications:**
- C:\Thesis\discovery\disc3.xml
  - Open

**Ranking Details:**
- C:\Thesis\ranking\rank1.xml
  - Open

**Qos Specifications:**
- C:\Thesis\qos\qos5.xml
  - Open

---

### MessagePart

**MessagePart Name:** input-1

**MessagePart Category:** Input

**Ontology URL:** `http://localhost/~kaarthik/LSDIS-ToyManufacturing.daml`

**Ontological Concept:** ToyIdentifier

**MessagePart Type:** String

---

**Add Message**  **Add Precondition**  **Add Effect**

**Collect**  **Update**  **Show Services**
Semantic Web Process Design

[Image of a software interface for Semantic Web Process Designer]

- View Process WSDL
- View Template
- Generate Process
- View BPEL Tree
- List Ontologies
- List Activities
- Interface Browser

Control Flow
Data Flow
Process Variables
Service Selection

Process Details
Add Web Services
Add Activity Interface
Add Semantic Activity Template

Update Activities
Hotel
List Services
Select Service
Save Details

<table>
<thead>
<tr>
<th>Business Name</th>
<th>Service Name</th>
<th>Operation Name</th>
<th>WSDL URL</th>
<th>Ranking Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BusinessNo6</td>
<td>HotelReservation</td>
<td>bookHotel</td>
<td><a href="http://lsdis.cs.uga.edu/proj/meteors/wsdl/Hotel">http://lsdis.cs.uga.edu/proj/meteors/wsdl/Hotel</a></td>
<td>0.666666666</td>
</tr>
<tr>
<td>BusinessSeven</td>
<td>Business7HotelService</td>
<td>bookHotel</td>
<td><a href="http://lsdis.cs.uga.edu/proj/meteors/wsdl/Hotel">http://lsdis.cs.uga.edu/proj/meteors/wsdl/Hotel</a></td>
<td>0.733333333</td>
</tr>
<tr>
<td>Demo1_NewBusiness2</td>
<td>TestHotelService2</td>
<td>bookHotel</td>
<td><a href="http://lsdis.cs.uga.edu/proj/meteors/wsdl/DontSellHotel">http://lsdis.cs.uga.edu/proj/meteors/wsdl/DontSellHotel</a></td>
<td>0.333333333</td>
</tr>
<tr>
<td>Demo1_NewBusiness3</td>
<td>TestHotelService3</td>
<td>bookHotel</td>
<td><a href="http://lsdis.cs.uga.edu/proj/meteors/wsdl/HotelService">http://lsdis.cs.uga.edu/proj/meteors/wsdl/HotelService</a></td>
<td>0.666666666</td>
</tr>
<tr>
<td>BusinessSeven</td>
<td>Business7HotelService</td>
<td>bookHotel</td>
<td><a href="http://lsdis.cs.uga.edu/proj/meteors/wsdl/Hotel">http://lsdis.cs.uga.edu/proj/meteors/wsdl/Hotel</a></td>
<td>0.733333333</td>
</tr>
<tr>
<td>Demo1_NewBusiness2</td>
<td>TestHotelService2</td>
<td>bookHotel</td>
<td><a href="http://lsdis.cs.uga.edu/proj/meteors/wsdl/DontSellHotel">http://lsdis.cs.uga.edu/proj/meteors/wsdl/DontSellHotel</a></td>
<td>0.333333333</td>
</tr>
<tr>
<td>Demo1_NewBusiness3</td>
<td>TestHotelService3</td>
<td>bookHotel</td>
<td><a href="http://lsdis.cs.uga.edu/proj/meteors/wsdl/HotelService">http://lsdis.cs.uga.edu/proj/meteors/wsdl/HotelService</a></td>
<td>0.666666666</td>
</tr>
</tbody>
</table>

[More information and functionalities visible in the software interface]
## Semantic Web Process Design

### Semantic Web Process Designer

<table>
<thead>
<tr>
<th>Source</th>
<th>From</th>
<th>Target</th>
<th>To</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assembly</td>
<td>(<a href="http://www.w3.org/2001/XMLSchema">http://www.w3.org/2001/XMLSchema</a>) : OutDate</td>
<td>RawMaterialDeliveryInterface (http:/...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expr</td>
<td>'AL-465'</td>
<td>RawMaterialDeliveryInterface (http:/...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Source Activity

- Service
- assemblyLine
- Output Messages
- (http://www.w3.org/2001/XMLSchema) : OutDate

### Target Activity

- RawMaterialDeliveryInterface

### Load Activities

- Input Messages
  - DeliveryLocation
- PickupDate
- PickupLocation
- DeliveryMeans
Semantic Web Process Design

Generate & Display BPEL Process
<?xml version="1.0" encoding="UTF-8"?>
  <partners>
    <partner name="caller" serviceLinkType="NS1:sampleConferenceArrangerSLT"/>
    <partner name="service-provider"/>
  </partners>
  <containers>
    <container messageType="NS1:arrange4ConferenceRequest" name="receive"/>
    <container messageType="NS2:getConferenceDetailsRequest" name="ConferenceDetails-request"/>
    <container messageType="NS2:getConferenceDetailsResponse" name="ConferenceDetails-response"/>
    <container messageType="NS3:bookHotelRequest" name="Hotel-request"/>
    <container messageType="NS3:bookHotelResponse" name="Hotel-response"/>
    <container messageType="NS4:bookAirTicketRequest" name="AirTicketTo-request"/>
    <container messageType="NS4:bookAirTicketResponse" name="AirTicketTo-response"/>
    <container messageType="NS4:bookAirTicketRequest" name="AirTicketReturn-request"/>
    <container messageType="NS4:bookAirTicketResponse" name="AirTicketReturn-response"/>
    <container messageType="NS1:arrange4ConferenceRequest" name="response"/>
  </containers>

  <sequence name="sequence-1">
    <receive container="receive" createInstance="yes" name="receive" operation="arrange4Conference" partner="caller" port="Service-Port"/>
    <assign name="ConferenceDetails"/>
    <copy>
      <from container="receive" part="ConferenceId"/>
      <to container="ConferenceDetails-request" part="ConferenceId"/>
    </copy>
  </sequence>
</process>
Ongoing Projects

- **SWAP**: [http://swap.semanticweb.org/](http://swap.semanticweb.org/)
  - Share knowledge effectively
  - Combination of Semantic Web and P2P

- **WonderWeb**: [http://wonderweb.man.ac.uk/](http://wonderweb.man.ac.uk/)
  - Development of a framework of techniques and methodologies that provide an engineering approach to the building and use of ontologies.
  - Development of a set of foundational ontologies covering a wide range of application domains.
  - Development of infrastructures and tool support that will be required by real world applications in the Semantic Web.
Ongoing Projects

- **DAML-S**: [http://www.daml.org/services/](http://www.daml.org/services/)
  - Set of ontologies to describe functionalities of web services

- **DAML-S Matchmaker**: [http://www-2.cs.cmu.edu/~7Esotagents/daml_Mmaker/daml-s_matchmaker.htm](http://www-2.cs.cmu.edu/~7Esotagents/daml_Mmaker/daml-s_matchmaker.htm)
  - Match service requestors with service providers
  - Semantic Matchmaking for Web Services Discovery

- **Web Service Composer**: [http://www.mindswap.org/~evren/composer/](http://www.mindswap.org/~evren/composer/)
  - Semi-automatic process for the dynamic composition of web services

  - WSDL, UDDI, SOAP
  - Business Process with BPEL4WS
Conclusions
Conclusions

- Semantic Web service Annotation and Discovery
  - Data semantics
  - Functional semantics
  - QoS Semantics
- Web processes vs. Semantic Web processes
  - BPEL4WS vs. DAML-S
- Web process composition
  - Web services semantic degree of integration
  - Data, Functional, and QoS similarity
- Web process QoS computation
  - QoS Models, techniques, and algorithms
Conclusions

- **Present Problems in Process Composition**
  - Static discovery of Web Services
  - Design/deployment-time binding of Web services
  - Process Composition is based on interfaces of participating services

- **Proposition**
  - Semantics is the enabler to address the problems of scalability, heterogeneity (syntactic and semantic), machine understandability faced by Web services

- **Semantics for Web Services**
  - Semantics can be applied to different layers of Web Services conceptual stack
  - Semantics for Web Services can be categorized into at least 4 different dimensions namely Data, Functional, Execution and Quality (QoS).
Conclusions

- Semantics can help address big challenges related to scalability, dynamic environments.
- But comprehensive approach to semantics will be needed:
  - Data/information, function/operation, execution, QoS
- Semantic (Web) principles and technology bring new tools and capabilities that we did not have in EAI, workflow management of the past

More at: http://lsdis.cs.uga.edu/proj/meteor/SWP.htm
Questions?
Web Resource for this tutorial (incl. latest version)

http://lsdis.cs.uga.edu/lib/presentations/SWSP-tutorial-resource.htm
References

DAML
http://www.daml.org/services/
http://www.daml.org/2001/03/daml+oil-index
References

Extensive related work at: IBM, Karlsruhe, U. Manchester, DAML-S (CMU, Stanford, UMD)

- [Sivashanmugam et al.-1] Adding Semantics to Web Services Standards
- [Verma et al.] MWSDI: A Scalable Infrastructure of Registries for Semantic Publication and Discovery of Web Services
- [Chandrasekaran et al.] Performance Analysis and Simulation of Composite Web Services
- [Cardoso et al.] Modeling Quality of Service for Workflows and Web Service Processes
- [Silver et al.] Modeling and Simulation of Quality of Service for Composition of Web Services
- [Paolucci et al.] Importing Semantic Web in UDDI

More at: http://lsdis.cs.uga.edu/SWP.htm