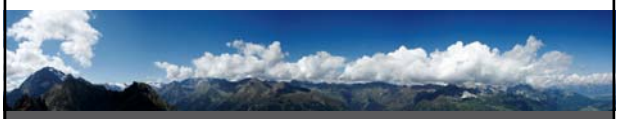
STI · INNSBRUCK

Semantic Web Services

OWL-S and Related Systems

Lecture IX – 14th May 2009

Srdjan Komazec



Where are we?

#	Date	Title
1	5th March	Introduction
2	12th March	Web Science
3	19th March	Service Science
4	26th March	Web Services (WSDL. SOAP, UDDI, XML)
5	2nd April	Web 2.0 and RESTful services
6	23rd April	WSMO
7	30th April	WSML
8	7th May	WSMX
9	**14th May**	**OWL-S and others**
10	28th May	WSMO-Lite, MicroWSMO
11	4th June	SWS Use Cases
12	18th June	seekda: the business point of view
13	25th June	Mobile services
14	2nd July	Exam Preparation

Agenda

- Motivation
- Comparison criteria
- OWL-S
  - Conceptual model, tool support and relation to WSMO
  - Criteria discussion
- METEOR-S
  - Annotation framework, discovery infrastructure, composition framework
  - WSDL-S
  - Relation to WSMO
  - Criteria discussion
- SWSF
  - Conceptual model, languages and relation to WSMO
  - Criteria discussion
- IRS – III
  - Features, framework and architecture
- Overall comparison

Motivation

- WSMO is not the only initiative aimed towards Semantic Web Services
- Other major approaches in the area are documented by recent W3C member submissions:
  - OWL-S
  - SWSF, and
  - WSDL-S/SAWSDL (i.e. METEOR-S)
- Other implementations of WSMO
  - IRS-III
- Goals of this lecture:
  - Explain principal characteristics of the approaches,
  - Give overview of the tools supporting the approach, and
  - Compare approach to WSMO.

Comparison Criteria

- Web compliance
  - Concepts of URI, namespaces, XML, decentralization of resources
- Ontology-based
  - Using ontologies as data models
- Strict decoupling
  - Resources are defined in isolation
- Centrality of mediation
  - Handling of the heterogeneities (data, underlying ontology, protocol and process)
- Ontological role separation
  - Epistemological differentiation of user desires and service offerings
- Description vs. implementation
  - Differentiation between the description elements and executable technologies
- Execution semantics
  - Formal execution semantics of reference implementations
- Service vs. Web services
  - Computational entity vs. actual value provided

OWL - S

OWL-S
Introduction

- OWL-S represents an ontology for the description of Semantic Web Services expressed in OWL.
- Has roots in the DAML Service Ontology (DAML-S, May 2001).
- Adopts existing Semantic Web recommendations (i.e. OWL)
- Maintains bindings to the Web Services world by linking to WSDL descriptions.
- An upper ontology for services
- W3C Member Submission November 22nd, 2004

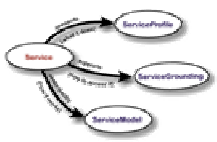


Figure 1 – OWL-S Conceptual Model

Figure taken from David Martin at al. OWL-S: Semantic Markup for Web Services, W3C Member Submission 22 November 2004

OWL-S
Service - Example

```
<rdf:RDF>
  <owl:Ontology rdf:about="">
    <rdfs:comment>
      This ontology represents the OWL-S service description for the BravoAir web service
    example.
    </rdfs:comment>
    <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.0/Service.owl"/>
    <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.0/BravoAirProfile.owl"/>
    <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.0/BravoAirProcess.owl"/>
  </owl:Ontology>

  <service:Service rdf:ID="BravoAir_ReservationAgent">
    <!-- Reference to the BravoAir Profile -->
    <service:presents rdf:resource="http://www.daml.org/services/owl-
    s/1.0/BravoAirProfile.owl#Profile_BravoAir_ReservationAgent"/>
    <!-- Reference to the BravoAir Process Model -->
    <service:describedBy rdf:resource="http://www.daml.org/services/owl-
    s/1.0/BravoAirProcess.owl#BravoAir_ReservationAgent_ProcessModel"/>
    <!-- Reference to the BravoAir Grounding -->
    <service:supports rdf:resource="http://www.daml.org/services/owl-
    s/1.0/BravoAirGrounding.owl#Grounding_BravoAir_ReservationAgent"/>
  </service:Service>
</rdf:RDF>
```

Example taken from http://www.daml.org/services/owl-s/1.0/examples.html

OWL-S
Service profile

- Expresses “what a service does” for
  - advertising purposes and
  - template for service requests.
- Specification of what functionality is provided by service through
  - inputs and outputs (OWL classes) and
  - preconditions and effects (format not fixed).
- Semantics of the conditions is not covered by the DL expressivity of OWL-S ontology.

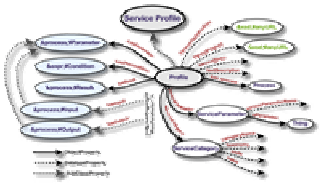


Figure 2 – OWL-S Service Profile

Figure taken from David Martin at al. OWL-S: Semantic Markup for Web Services, W3C Member Submission 22 November 2004

OWL-S
Service profile – properties 1

- Service Profile
  - **presents -** describes a relation between an instance of service and an instance of profile, it basically says that the service is described by the profile
  - **presentedBy -** is the inverse of presents; it specifies that a given profile describes a service
- Service Name, Contacts and Description
  - **serviceName -** refers to the name of the service that is being offered. It can be used as an identifier of the service
  - **textDescription -** provides a brief description of the service. It summarizes what the service offers, it describes what the service requires to work, and it indicates any additional information that the compiler of the profile wants to share with the receivers
  - **contactInformation -** provides a mechanism of referring to humans or individuals responsible for the service (or some aspect of the service). The range of this property is unspecified within OWL S but can be restricted to some other ontology, such as FOAF, VCard, or the now depreciated Actor class provided in previous versions of OWL S

OWL-S
Service profile – properties 2

- Functionality Description
  - **hasParameter -** ranges over a Parameter instance of the Process ontology. Note that the Parameter class models our intuition that Inputs and Outputs (which are kinds of Parameters) are both involved in information transformation and therefore they are different from Preconditions and Effects. As a consequence, we do not expect this class to be instantiated. It's role is solely making domain knowledge explicit.
  - **hasInput -** ranges over instances of Inputs as defined in the Process Ontology
  - **hasOutput -** ranges over instances of type Output, as defined in the Process ontology
  - **hasPrecondition -** specifies one of the preconditions of the service and ranges over a Precondition instance defined according to the schema in the Process ontology
  - **hasResult -** specifies one of the results of the service, as defined by the Result class in the Process ontology. It specifies under what conditions the outputs are generated. Furthermore, the Result specifies what domain changes are produced during the execution of the service
- Profile Attributes
  - **serviceParameter -** is an expandable list of properties that may accompany a profile description. The value of the property is an instance of the class ServiceParameter
  - **serviceCategory -** refers to an entry in some ontology or taxonomy of services. The value of the property is an instance of the class ServiceCategory

OWL-S
Service profile – properties 3

- Service Parameter
  - **serviceParameterName -** is the name of the actual parameter, which could be just a literal, or perhaps the URI of the process parameter (a property)
  - **sParameter -** points to the value of the parameter within some OWL ontology
- ServiceCategory
  - **categoryName -** is the name of the actual category, which could be just a literal, or perhaps the URI of the process parameter (a property)
  - **taxonomy -** stores a reference to the taxonomy scheme. It can be either a URI of the taxonomy, or a URL where the taxonomy resides, or the name of the taxonomy or anything else
  - **value -** points to the value in a specific taxonomy There may be more than one value for each taxonomy, so no restriction is added here
  - **code -** to each type of service stores the code associated to a taxonomy
- Service Type and Product
  - **serviceClassification -** defines a mapping from a Profile to an OWL ontology of services, such as an OWL specification of NAICS
  - **serviceProduct -** defines a mapping from a Profile to an OWL ontology of products, such as an OWL specification of UNSPSC

OWL-S
Service profile – example

```
<rdf:RDF>
   <owl:Ontology rdf:about="">
     <rdfs:comment>BravoAir Example for OWL-S Profile description</rdfs:comment>
     <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.0/Service.owl"/>
     ...
   </owl:Ontology>
  <profile:serviceParameter>
    <addParam:GeographicRadius rdf:ID="BravoAir-geographicRadius">
      <profile:serviceParameterName>BravoAir Geographic Radius</profile:serviceParameterName>
      <profile:sParameter rdf:resource="http://www.daml.org/services/owl-
      s/1.0/Country.owl#UnitedStates"/>
    </addParam:GeographicRadius>
  </profile:serviceParameter>

  <profile:serviceCategory>
    <addParam:NAICS rdf:ID="NAICS-category">
      <profile:value>Airline reservation services </profile:value>
      <profile:code>561599</profile:code>
    </addParam:NAICS>
  </profile:serviceCategory>
  ...
  <profile:hasInput rdf:resource="http://www.daml.org/services/owl-
    s/1.0/BravoAirProcess.owl#DepartureAirport_In"/>
  <profile:hasOutput rdf:resource="http://www.daml.org/services/owl-
    s/1.0/BravoAirProcess.owl#AvailableFlightItineraryList_Out"/>
</rdf:RDF>
```

Example taken from http://www.daml.org/services/owl-s/1.0/examples.html

OWL-S
Service model

- Exposes "how a service works" to enable invocation, composition, monitoring, recovery, etc.
- The model views interaction of the service as a process.
- Distinguishes atomic, simple and composite processes.
- Semantics of the workflow constructs is not expressible in the DL underlying OWL.

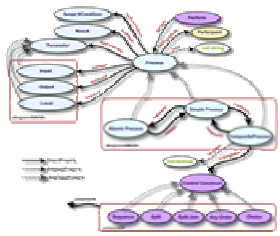


Figure 3 – OWL-S Service Process Model

Figure taken from David Martin at al. OWL-S: Semantic Markup for Web Services, W3C Member Submission 22 November 2004

OWL-S
Service model – processes

- Atomic processes
  - directly invocable (by passing them the appropriate messages),
  - no subprocesses and execute in a single step,
  - they take an input message, do something, and then return their output message,
  - provided a grounding that enables a service requester to construct messages to the process from its inputs and deconstruct replies
- Simple processes
  - not invocable and are not associated with a grounding,
  - conceived of as having single-step executions
  - used as elements of abstraction
    - used to provide a view of (a specialized way of using) some atomic process
    - simplified representation of some composite process (for purposes of planning and reasoning)
- Composite processes
  - decomposable into other (non-composite or composite) processes,
  - their decomposition can be specified by using control constructs such as Sequence and If-Then-Else
  - a composite process is not a behavior a service *will* do, but a behavior (or set of behaviors) the client *can* perform by sending and receiving a series of messages

OWL-S
Service model – control constructs 1

- **Sequence**
  - A list of control constructs to be done in order
- **Split**
  - The components of a Split process are a bag of process components to be executed concurrently. Split completes as soon as all of its component processes have been scheduled for execution
- **Split+Join**
  - Here the process consists of concurrent execution of a bunch of process components with barrier synchronization. That is, Split+Join completes when all of its components processes have completed. With Split and Split+Join, we can define processes that have partial synchronization (e.g., split all and join some subbag).
- **Any-Order**
  - Allows the process components (specified as a bag) to be executed in some unspecified order but not concurrently. Execution and completion of all components is required. The execution of processes in an Any-Order construct cannot overlap, i.e. atomic processes cannot be executed concurrently and composite processes cannot be interleaved. All components must be executed. As with Split+Join, completion of all components is required

OWL-S
Service model – control constructs 2

- **Choice**
  - calls for the execution of a single control construct from a given bag of control constructs (given by the components property). Any of the given control constructs may be chosen for execution.
- **If-Then-Else**
  - control construct that has properties ifCondition, then and else holding different aspects of the If Then Else. Its semantics is intended as ``Test If condition; if True do Then, if False do Else.'' (Note that the class Condition, which is a placeholder for further work, will be defined as a class of logical expressions.)
- **Iterate**
  - makes no assumption about how many iterations are made or when to initiate, terminate, or resume. The initiation, termination or maintenance condition could be specified with a whileCondition or an untilCondition
- **Repeat-While and Repeat-Until**
  - Both of these iterate until a condition becomes false or true, following the familiar programming language conventions

OWL-S
Service model – example

```
<rdf:RDF>
  <process:ProcessModel rdf:ID="BravoAir_ReservationAgent_ProcessModel">
    <process:hasProcess rdf:resource="#BravoAir_Process"/>
    <service:describes rdf:resource="http://www.daml.org/services/owl-
      s/1.0/BravoAirService.owl#BravoAir_ReservationAgent"/>
  </process:ProcessModel>

  <process:CompositeProcess rdf:ID="BravoAir_Process">
    <process:composedOf>
      <process:Sequence>
        <process:components rdf:parseType="Collection">
          <process:AtomicProcess rdf:about="#GetDesiredFlightDetails"/>
          <process:AtomicProcess rdf:about="#SelectAvailableFlight"/>
          <process:CompositeProcess rdf:about="#BookFlight"/>
        </process:components>
      </process:Sequence>
    </process:composedOf>
  </process:CompositeProcess>
```

Example taken from http://www.daml.org/services/owl-s/1.0/examples.html

OWL-S
Service model – example

```
  <process:CompositeProcess rdf:ID="BookFlight">
    <process:composedOf>
      <process:Sequence>
        <process:components rdf:parseType="Collection">
          <process:AtomicProcess rdf:about="#Login"/>
          <process:AtomicProcess rdf:about="#ConfirmReservation"/>
        </process:components>
      </process:Sequence>
    </process:composedOf>
  </process:CompositeProcess>

  <process:AtomicProcess rdf:ID="LogIn">
    <process:hasInput rdf:resource="#AcctName_In"/>
    <process:hasInput rdf:resource="#Password_In"/>
  </process:AtomicProcess>

  <process:Input rdf:ID="AcctName_In">
    <process:parameterType rdf:resource="http://www.daml.org/services/owl-
    s/1.0/Concepts.owl#AcctName"/>
  </process:Input>

  <process:Input rdf:ID="Password_In">
    <process:parameterType rdf:resource="http://www.daml.org/services/owl-
    s/1.0/Concepts.owl#Password"/>
  </process:Input>
</rdf:RDF>
```


OWL-S
Service grounding

- Maps the constructs of the process model to detailed specifications of message formats, protocols and so forth.
- Mapping of the atomic processes to WSDL operations and their I/Os to WSDL messages is supported.
- Mappings might have XSLT transformations attached to solve the lifting/lowering problem between OWL and XML Schema.
- Other grounding mechanisms can be supported
- For example take a look at http://www.daml.org/services/owl-s/1.0/BravoAirGrounding.owl

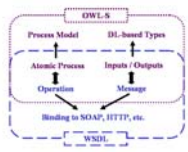


Figure 4 – Mapping between OWL-S and WSDL

Figure taken from David Martin at al. OWL-S: Semantic Markup for Web Services, W3C Member Submission 22 November 2004

OWL-S
Tool support

- Reasoning
  - OWL DL reasoners like Pellet and FaCT/FaCT++,
  - Rule oriented extensions and embedded SWRL are used to overcome limitations of OWL DL, as well as reasoners Racer and Kaon2, and
  - Possibility to use metareasoning.
- Discovery
  - Match making relies on the explicit specification of inputs and outputs,
  - Ontological relationships between the I/Os of "template" and a candidate services are compared,
  - A set of five filters classify structural relationships between the template and candidate I/Os
  - OWLS MX matchmaker implements this classification and combines it with syntactic match
  - There is no full usage of IOPE based matchmaking, neither discovery approach which takes into account behavioral aspects of the service

OWL-S
Tool support

- Choreography and Orchestration
  - OWL S assumes invocation of services as atomic actions (choreography is not modeled)
  - *Process-model* allows services to be attached to the atomic but also composite processes thus allowing hierarchical control flow oriented decomposition (i.e. ordering over atomic processes)
  - OWL S VM is executes the atomic processes up to the completion of the composite process description.
  - OWL S VM fulfills the basic role of an orchestration engine.
- Mediation
  - No first class support for data and process mediation.
- Composition
  - Abstraction of service interactions to atomic ones and its simple model of composite processes are good fit to existing (AI planning based) techniques.

OWL-S
Relationship to WSMO

- Underlying specification
  - OWL S is specified using OWL, and WSMO uses abstract MOF model.
- Language unification
  - OWL S needs to combine OWL with more expressive languages (e.g. for expressing conditions and workflow constructs) while WSMO provides a single unified language framework.
- Conceptual model similarities
  - OWL S Service Profile is close to the capability of a service/goal in WSMO.
    - WSMO distinguishes requester's and provider's view
  - OWL S Process Model is conceptually similar to the WSMO service/goal interfaces.
    - WSMO distinguishes between the external and internal behavior
- OWL-S lacks mediation facilities
- Grounding similarities
  - Both approaches adopt similar ideas with respect to binding to WSDL.
    - It is a top level concept in OWL-S and not in WSMO

OWL-S
Criteria discussion

- Web compliance
  - Supports URI, namespaces, XML
- Ontology-based
  - Based on OWL
- Strict decoupling
  - Ontologies and service descriptions can be specified independently
- Centrality of mediation
  - No support for mediation
- Ontological role separation
  - Doesn't have notion of user desires
- Description vs. implementation
  - OWL S is a description framework based on appropriate formalisms in order to provide concise semantic descriptions
- Execution semantics
  - Doesn't have it
- Service vs. Web services

METEOR - S

METEOR-S
Introduction

- METEOR-S project defines semantics for the complete lifecycle of Semantic Web processes including annotation, discovery, composition and enactments of Web Services.
- It is strongly coupled with existing Web Services standards, thus extending them with semantics.
- The project focuses on
  - semantic annotation of Web Services,
    - METEOR-S Web Service Annotation Framework (MWSAF),
  - semantics based discovery of Web Services,
    - METEOR-S Web Service Discovery Infrastructure (MWSDI),
  - composition encompassing data mediation
    - METEOR-S Web Service Composition Framework (MWSCF).

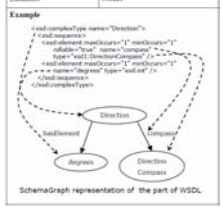
METEOR-S
Annotation Framework (MWSAF)

- Framework for semiautomatic annotation of Web Services' semantics addressing four different aspects:
  - Semantics of the inputs and outputs of Web Services,
  - Functional semantics ("what service does"),
  - Execution semantics to support verification of the correctness of the Web Service executions, and
  - Inclusion of information regarding the quality of service (performance, costs, …).
- Semiautomatic annotation is based on
  - Transformation of both XML Schema part of Web Service definitions and ontologies into a common representation – SchemaGraph, and
  - Matching algorithms which compute "match score" between the SchemaGraph elements.
- MWSAF comprises:
  - Ontology store – ontologies to be used during Web Service annotation,
  - Matcher library – algorithm implementations for linguistic and structural matching and
  - Translator library – SchemaGraph generation procedures.

METEOR-S
Annotation Framework (MWSAF) - Example

XML schema Construct	SchemaGraph representation
ComplexType	Node
Elementary XML Data Type Element defined under complexType	Node and an Edge between complexType node and this node with name "hasElement"
ComplexType XML Data Type Element defined under complexType	Edge
SimpleType	Node
Values defined for simple types	Node and edge between simpleType and this node with name "hasValue"
Elements	Nodes

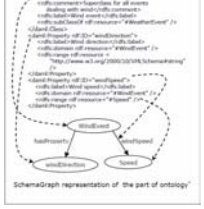
Example



SchemaGraph representation of the part of WSDL

Ontology representation	SchemaGraph representation
Class	Node
Property with basic datatypes as range (Attribute)	Node with edge joining it to the class with name "hasProperty"
Property with other class as range (Attribute)	Edge between the two class nodes
Instance	Node with edge joining it to the class with name "hasInstance"
Class – subclass relationship	Edge between class node to subclass node with name "hasSubClass"

Example



SchemaGraph representation of the part of ontology

Taken from Abhijit Patil et al. METEOR-S Web service Annotation Framework, Proceeding of the World Wide Web Conference, July 2004

METEOR-S
WSDL-S and SAWSDL

- Lightweight approach to associating semantic annotations with Web Services which builds upon existing standards.
- Relies on the extensibility of WSDL (i.e. XML) to add semantic annotations in the form of URI references to external models to the interface, operation and message constructs.
- WSDL-S was superseded by SAWSDL which is a restricted and homogenized version of WSDL-S.

```
<wsdl:description
  targetNamespace="http://www.w3.org/2002/ws/sawsdl/spec/wsdl/order#"
  xmlns=http://www.w3.org/2002/ws/sawsdl/spec/wsdl/order#
  xmlns:wsdl=http://www.w3.org/ns/wsdl
  xmlns:xs=http://www.w3.org/2001/XMLSchema
  xmlns:sawsdl="http://www.w3.org/ns/sawsdl">
  <wsdl:types>
    <xs:schema targetNamespace=http://www.w3.org/2002/ws/sawsdl/spec/wsdl/order#
               elementFormDefault="qualified">
      <xs:element name="OrderRequest"
        sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/purchaseorder#OrderRequest"
        sawsdl:loweringSchemaMapping="http://www.w3.org/2002/ws/sawsdl/spec/mapping/RDFOnt2Request.xml">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="customerNo" type="xs:integer" />
            ...
</wsdl:description>
```


METEOR-S
Discovery Infrastructure (MWSDI)

- Discovery
  - METEOR-S Web Services Discovery Infrastructure (MWSDI) attempts to enhance existing WS discovery infrastructure by using semantics,
  - Unified access to a large number of 3rd party registries,
  - Scalability and flexibility are achieved by using P2P techniques.

Enhances the framework with semantics:
- Registries ontologies, and
- Domain-specific ontologies.

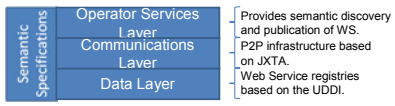


Figure 5 – Layered Architecture of MWSDI

METEOR-S
Composition Framework

- Composition
  - METEOR-S Web Service Composition Framework (MWSCF) aims to increase the flexibility of WS compositions by relying on semantic process templates.
  - Template defines process in terms of semantically defined activities (BPEL control flow constructs with activities).
  - Template is used to generate executable processes by binding the semantically defined activities to concrete WS.

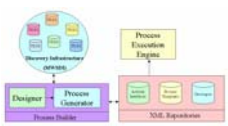


Figure 5 – Composition framework

Figure taken from Sivashanmugam, K., et al. Framework for Semantic Web Process Composition, Special Issue of the International Journal of Electronic Commerce (IJEC), Eds: Christoph Bussler, Dieter Fensel, Norman Sadeh, Feb 2004

METEOR-S
Composition Framework - Steps

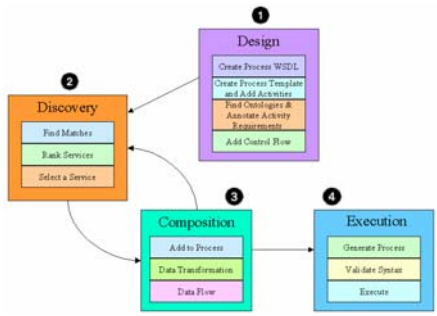


Figure taken from Sivashanmugam, K., et al. Framework for Semantic Web Process Composition, Special Issue of the International Journal of Electronic Commerce (IJEC), Eds: Christoph Bussler, Dieter Fensel, Norman Sadeh, Feb 2004

METEOR-S
WSDL-S relationship to WSMO

- With its minimalistic approach can be viewed as orthogonal to WSMO,
- It can reference WSMO descriptions from its annotations,
- All the WSDL-S referenced concepts can be annotated in WSML/WSMO.
- WSDL-S doesn't clearly establish how requests should be defined
  - Usage of different languages hinder possibility to formally define requests, queries or notions of a "match" between service requests and service descriptions.
- Other WSMO supported notions such as nonfunctional properties and behavioral/interface descriptions should be addressed by the initiative in the same manner (i.e. extending WS-Policy, BPEL or WS-CDL with lightweight notions).
- WSDL-S has a strong industrial point
  - WSMO/L/X working groups are looking at closer alignment with industry standards and adoption of WSDL S as a possible future grounding formalism for WSMO.
- WSDL-S has been suppressed by SAWSDL.

METEOR-S
Criteria discussion

- Web compliance
  - Supports URI, namespaces, XML
- Ontology-based
  - Based on DAML and RDF S
- Strict decoupling
  - Ontologies and service descriptions can be specified independently, connected through SAWSDL annotations
- Centrality of mediation
  - No support for mediation
- Ontological role separation
  - Doesn't have notion of user desires
- Description vs. implementation
  - Follows a much more technology centered approach, not providing a conceptual model for the description of services and their related aspects
- Execution semantics
  - Doesn't have it
- Service vs. Web services

SWSF

SWSF
Introduction

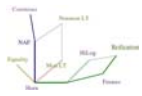
- Semantic Web Services Framework (SWSF) has roots in OWL-S and the Process Specification Language (PSL).
- Based on two major components:
  - Ontology (Conceptual Model)
    - **F**irst-Order **L**ogic **O**ntology for **W**eb **S**ervices (FLOWS), and
    - **R**ules **O**ntology for **W**eb **S**ervices (ROWS)
  - Language
    - **SWSL-FOL, and**
    - **SWSL-Rules.**

SWSF
Conceptual Model

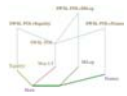
- Semantic Web Service Ontology (SWSO)
  - Influenced by OWL-S (shares the same 3 concepts),
  - Extension/refinement of OWL-S
    - Underlying language (SWSL) is more expressive, and
    - Richer behavioral process model based on PSL.
- Two independent formalizations of the conceptual model
  - FLOWS (First-order Logic Ontology for Web Services), and
    - Relies on the semantic of SWLS-FOL
  - ROWS (Rule Ontology for Web Services)
    - Relies on the semantics of SWLS-Rules.
- In order to describe behavior of the service based on PSL approach two fundamental elements are added:
  - Structured notion of atomic processes, and
  - Infrastructure for specifying various forms of data flow.
- The process ontology is made up of six parts – divided according to their expressivity.

SWSF
Languages

- Describe concept of Web Services but also individual services.
- Languages are compliant with Web principles
  - Use of URI, integration with XML built-in types, XML-compatible namespaces, import mechanism.
- Languages are layer with increased modeling power.
- Two variants:
  - SWSL-FOL
    - First-order logic including features from HiLog and F-Logic.
  - SWSL-Rules
    - Logic programming language including features from courteous-logic programs, HiLog, and F-Logic,
    - Supports service-related tasks such as discovery, contracting, and policy specification.



The Layered Structure of SWSL-Rules



Layers of SWSL-FOL and Their Relationship to SWSL-Rules

Figures taken from Steve Battle at al. Semantic Web Services Language (SWSL) W3C Member Submission 9 September 2005

SWSF
Relationship to WSMO

- From the conceptual model point of view the same consideration apply like in the case of OWL-S.
- Many of the deficiencies of OWL-S language have been overcome in SWSL:
  - Not being bound to the restricted expressivity of DL,
  - Providing rigid definitions of semantic aspects of conditions and dynamic aspects of SWS by reuse of PSL, and
  - Providing an single semantic language framework.
- Lack of tools

SWSF
Criteria discussion

- Web compliance
  - Supports URI, namespaces, XML
- Ontology-based
- Strict decoupling
  - Not supported
- Centrality of mediation
  - No support for mediation
- Ontological role separation
  - Doesn't have direct notion of user desires
- Description vs. implementation
  - Provides serialization/deserialization mechanisms to/from WSDL
- Execution semantics
  - Doesn't have it
- Service vs. Web services

IRS - III

Internet Reasoning Service (IRS-III)

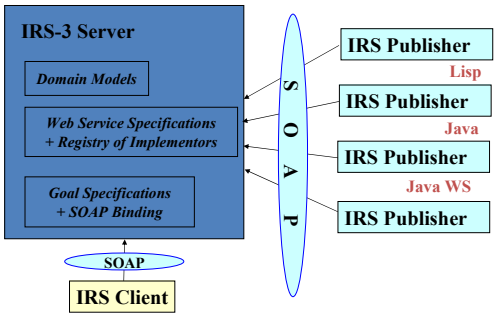
- IRS III is a framework and implemented infrastructure which supports the creation of semantic web services according to the WSMO ontology
- Builds upon the previous version by incorporating and extending the WSMO ontology within the IRS-III server, browser and API.

Internet Reasoning Service (IRS-III)

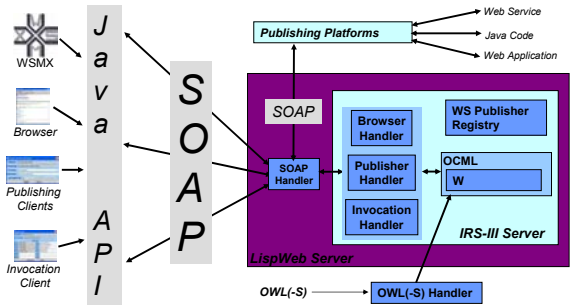
- Principles underlying IRS-III
  - Ontological separation of User and Web Service Contexts
  - Capability Based Invocation
  - Ease of Use
  - One Click Publishing
  - Agnostic to Service Implementation Platform
  - Connected to External Environment
  - Open
  - Complete Descriptions
  - Inspectable
  - Interoperable with SWS Frameworks and Platforms

Some features of IRS III

- Based on SOAP messaging standard
- Provides Java API for client applications
- Provides built-in brokering and service discovery support
- Provides *capability-centred* service invocation
- Publishing support for variety of platforms
  - Java, Lisp, Web Applications, Java Web Services
- Enables publication of 'standard code'
  - Provides clever wrappers
  - One-click publishing of web services
- Integrated with standard Web Services world
  - Semantic web service to IRS
  - 'Ordinary' web service



IRS-III Architecture



Figures taken from John Domingue at al. Demo of IRS-III: A Platform and Infrastructure for Creating WSMO-based Semantic Web Services, Demo from ISWC 2004

IRS-III/WSMO Difference

- Underlying language OCML
- Goals have inputs and outputs
- IRS-III broker finds applicable web services via mediators
  - Used mediator within WS capability
  - Mediator source = goal
- Web services have inputs and outputs 'inherited' from goal descriptions
- Web service selected via assumption (in capability)

Overall Comparison

	WSMO	OWL-S	METEOR-	SWSF
Web compliance	√	√	√	√
Ontology-based	√	√	√	√
Strict decoupling	√	√	√	-
Centrality of mediation	√	-	-	-
Ontological role separation	√	-	-	-
Description vs. implementatio	√	√	-	√
Execution semantics	√	-	-	-
Services vs. Web services	√	√	√	√

Learning material

- Dieter Fensel, Mick Kerrigan, Michal Zaremba (Eds.), Implementing *Semantic Web Services: The SESA Framework*. Springer-Verlag, 2008.
- David Martin, et al., *OWL-S: Semantic Markup for Web Services*, W3C Member Submission 22 November 2004, http://www.w3.org/Submission/OWL-S.
- Joel Farrell and Holger Lausen, *Semantic Annotations for WSDL and XML Schema*, W3C Recommendation 28 August 2007, http://www.w3.org/TR/sawsdl
- Rama Akkiraju et al., *Web Service Semantics - WSDL-S*, W3C Member Submission 7 November 2005, http://www.w3.org/Submission/WSDL-S
- Dieter Fensel, Holger Lausen, Axel Polleres, Jos de Bruijn, Michael Stollberg, Dumitru Roman, John Domingue, *Enabling Semantic Web Services: The Web Service Modeling Ontology*, Springer-Verlag, 2007.
- Steve Battle et al., *Semantic Web Services Framework (SWSF)*, W3C Member Submission 9 September 2005, http://www.w3.org/Submission/SWSF

Next Lecture

#	Date	Title
1	5th March	Introduction
2	12th March	Web Science
3	19th March	Service Science
4	26th March	Web Services (WSDL. SOAP, UDDI, XML)
5	2nd April	Web 2.0 and RESTful services
6	23rd April	WSMO
7	30th April	WSML
8	7th May	WSMX
9	14th May	OWL-S and others
10	**28th May**	**WSMO-Lite, MicroWSMO**
11	4th June	SWS Use Cases
12	18th June	seekda: the business point of view
13	25th June	Mobile services
14	2nd July	Exam Preparation

Questions?