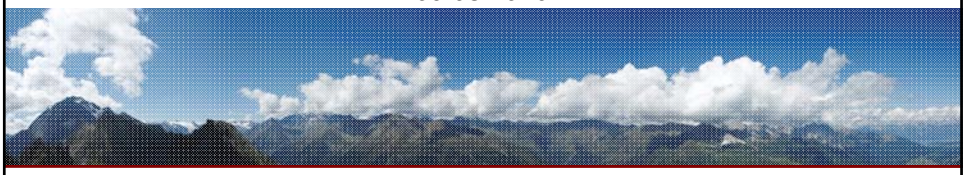
Semantic Web Services
SS 2016

OWL-S and others

Anna Fensel

09.05.2016



© Copyright 2010 Dieter Fensel and Srdjan Komazec

Where are we?

STI · INNSBRUCK

#	Title
1	Introduction
2	Web Science + TourPack project (separate slideset)
3	Service Science
4	Web services
5	Web2.0 services + ONLIM APIs (separate slideset)
6	Semantic Web
7	Semantic Web Service Stack (WSMO, WSML, WSMX)
8	**OWL-S and the others**
9	Semantic Services as a Part of the Future Internet and Big Data Technology
10	Lightweight Annotations
11	Linked Services
12	Applications
13	Mobile Services

Outline

STI · INNSBRUCK

- Motivation
- Technical solution
  - Comparison approach
    - Framework description
    - Comparison criteria
  - Evaluated Semantic Web Service (SWS) frameworks
    - OWL-S
    - METEOR-S
    - SWSF
  - Evaluated Semantic Execution Environment (SEE) implementation
    - IRS – III
- Illustration by larger example
- Summary
- References

3

STI · INNSBRUCK

Motivation

4

Motivation

STI · INNSBRUCK

- 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,
  - METEOR-S, and
  - SWSF.
- Other implementations of WSMO
  - IRS-III
- Goals of this lecture:
  - Explain principal characteristics of the approaches,
  - Exemplify their usage,
  - Give overview of the tools support,
  - Compare approach to WSMO (for IRS-III to WSMX), and
  - Evaluate approach in contrast to the SWS framework requirements.

5

STI · INNSBRUCK

Technical Solution
Framework description

6

Comparison Approach
Framework Description

STI · INNSBRUCK

- Overall approach
  - Describes principal characteristics of evaluated framework
- Conceptual model
  - Explains underlying model adopted by the framework
- Example
  - Illustrates the framework usage through an example
- Tool support
  - Overviews existing tools developed to support the framework
- Relation to WSMO
  - Draws parallel between WSMO and evaluated framework

7

STI · INNSBRUCK

Technical solution
Comparison criteria

8

Comparison Approach
Comparison Criteria

STI · INNSBRUCK

- SWS are basically combining Semantic Web technologies and Web services.
- Thus SWS frameworks need to integrate
  - basic Web design principles,
  - design principles defined for the Semantic Web, and
  - design principles for distributed, service-oriented computing on the Web.
- A SWS framework should answer to the following requirements[1]:
  - Web compliance
  - Ontology-based
  - Strict decoupling
  - Centrality of mediation
  - Ontological role separation
  - Description vs. implementation
  - Execution semantics
  - Service vs. Web service

[1] Fensel, Dieter; Kerrigan, Mick; Zaremba, Michal (Eds.). Implementing Semantic Web Services: The SESA Framework. Springer, 2008

9

Comparison Approach
Comparison Criteria – Web Compliance

STI · INNSBRUCK

- What is it?
  - A set of simple and flexible concepts forming the foundation of Web such as Universal Resource Identifier (URI), namespaces, etc.
- Why is it important?
  - Reliance on the already established concepts such as URIs enables seamless integration of pre-existing Web resources and application of proven and tested tools and libraries.
- What is expected from the framework?
  - A SWS framework should integrate with the current Web, thus it should reuse/rely on the basic Web concepts.

10

Comparison Approach
Comparison Criteria – Ontology-based

STI · INNSBRUCK

- What is it?
  - Ontologies are formal and explicit specifications of shared conceptualizations.
  - An ontology-based system is a system which adopts ontologies as a founding format to express data and processes which the system deals with.
- Why is it important?
  - Ontologies are allowing enhanced information processing and automatic resolution of heterogeneity issues between resources.
- What is expected from the framework?
  - Usage of ontologies as the data model throughout framework.
  - All resource descriptions and data exchanged are ontologically described.

11

Comparison Approach
Comparison Criteria – Strict Decoupling

STI · INNSBRUCK

- What is it?
  - Decoupling represents separation of resources which should not depend on each other.
  - Size, diversity and unanticipated usage of Web resources are advocating towards their strict decoupling.
- Why is it important?
  - On the Web resources are developed in isolation from one another.
- What is expected from the framework?
  - Resources should be strictly decoupled from one another.
  - Semantic descriptions are developed in isolation without regard for their possible usage or interaction with other resources.

12

Comparison Approach
Comparison Criteria – Centrality of Mediation

STI · INNSBRUCK

- What is it?
  - Mediation is an activity which reconciles heterogeneities between parties.
- Why is it important?
  - Decoupling of resources requires existence of a mechanism for resolving heterogeneity issues between them.
  - Mediation performs the role of resolving potential mismatches between resources that occur at the data, protocol and process level.
- What is expected from the framework?
  - Mediators should be top-level elements of the framework.
  - It should be easy to connect mediators to the other elements.

13

Comparison Approach
Comparison Criteria – Ontological Role Separation

STI · INNSBRUCK

- What is it?
  - Consumers and providers of Web Services are loosely coupled.
  - Consumption of a Web Service functionality can be cannot be expected neither enforced.
- Why is it important?
  - The contexts within which requesters and providers of services exist can be very different from one another.
- What is expected from the framework?
  - A framework should differentiate between the requirements that a given requester has and the functionality that service providers are willing to offer.
  - The framework should stress the difference between the goals and Web Services.

14

Comparison Approach
Comparison Criteria – Description vs. Implementation

STI · INNSBRUCK

- What is it?
  - Difference between description of functionality and its implementation.
  - Service description is separated from the way the service is implemented and offered.
- Why is it important?
  - A firm distinction between the description of elements of Semantic Web Services and executable technologies should be present.
  - It allows decoupling from the current implementation technologies and possibility to ground descriptions to different underlying standards.
- What is expected from the framework?
  - A framework should aim at providing ontological description model for the elements and to be compliant with the executable technologies.

15

Comparison Approach
Comparison Criteria – Execution Semantics

STI · INNSBRUCK

- What is it?
  - It is a formal description of the operational behavior of the system.
  - By separating the description of the system behavior from the system implementation greater flexibility can be achieved.
- Why is it important?
  - It provides a mechanism to test, simulate and verify the SWS specification, identify anomalies such as deadlocks, livelocks and tasks that are never reached.
  - It provides sound foundation to build an engine being able to execute created models.
  - It improves or even rules out ambiguities in model understanding by humans.
- What is expected from the framework?
  - Framework should provide means to specify and run formal execution semantics.

16

Comparison Approach
Comparison Criteria – Service vs. Web Service

STI · INNSBRUCK

- What is it?
  - A clear distinction between Services and Web Services.
  - A Web Service is a computational entity that is able to provide some functionality to the user by invoking it.
  - A service is the actual value provided by the invocation.
- Why is it important?
  - Although services are what counts for the consumers their value may be subjective and context-dependent.
  - A SWS framework should provides means to describe Web Services.
- What is expected from the framework?
  - A framework should be designed as a means to describe Web Services and not to replace services.
  - Actual value is provided by Web Service invocations.

17

STI · INNSBRUCK

Technical solution
OWL – S

18

OWL-S
Overall approach

STI · INNSBRUCK

- OWL-S represents an upper ontology for the description of Semantic Web Services expressed in OWL.
  - It has its roots in the DAML Service Ontology (DAML-S).
  - It adopts existing Semantic Web recommendations (i.e. OWL).
  - It maintains bindings to the Web Services world by linking to WSDL descriptions.
- W3C Member Submission

19

OWL-S
Conceptual Model – Service Class

STI · INNSBRUCK

- The class **Service**
  - provides an organizational point of reference for a declared Web service.
- The class has three properties:
  - **presents**
    - Defines what service does.
    - Points to the ServiceProfile instance.
  - **supports**
    - Defines how to access the described service.
    - Points to the ServiceGrounding instance.
  - **describedBy**
    - Defines how the service works.
    - Points to the ServiceModel instance.

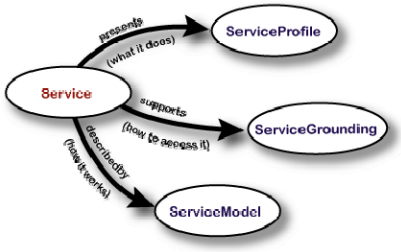


Figure 1 – OWL-S Conceptual Model

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

20

OWL-S
Conceptual Model – Service Profile Class

STI · INNSBRUCK

- Expresses “what a service does” for
  - Advertising purposes - used to express the service functional and non functional properties, and
  - Template for service requests.
- Specification of **functionality** provided by a service is given through:
  - requirements that the service requester must satisfy to use the service successfully.
  - Inputs and outputs (OWL classes) and
  - Preconditions and results (format not fixed).
- **Non-functional properties** of a service are defined through:
  - Instances of the ServiceCategory class
    - an entry in some ontology or taxonomy of services.
  - Instances of the ServiceParameter class
    - expandable list of arbitrary properties that may accompany a profile description.
  - A set of simple value attributes such as serviceName, textDescription, serviceProduct, serviceClassification (explained later).

21

OWL-S
Conceptual Model – Service Profile Class

STI · INNSBRUCK

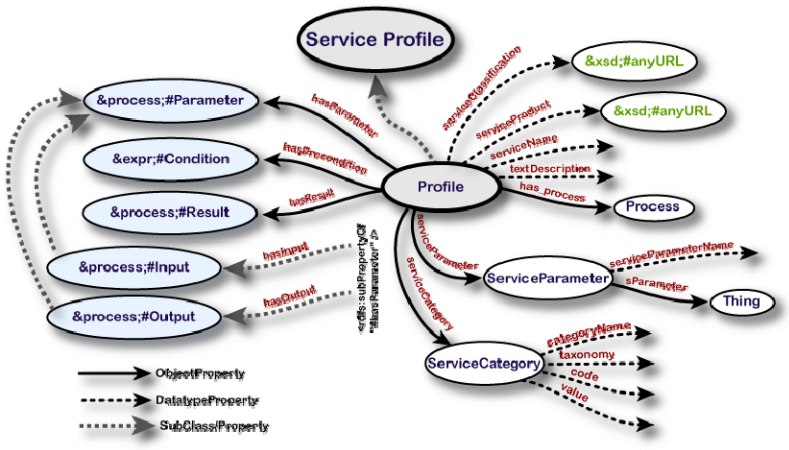


Figure 2 – OWL-S Service Profile

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

22

OWL-S
Conceptual Model – Service Profile Class

STI · INNSBRUCK

- Functional properties
  - **hasParameter -** ranges over a Parameter instance from the Service Model[2] 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 Service Model[2] ontology.
  - **hasOutput -** ranges over instances of type Output, as defined in the Service Model[2] ontology.
  - **hasPrecondition -** specifies one of the preconditions of the service and ranges over a Precondition instance defined according to the schema in the Service Model[2] ontology.
  - **hasResult -** specifies one of the results of the service, as defined by the Result class in the Service Model[2] 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

[2] The Service Model ontology is explained in the upcoming slides.

23

OWL-S
Conceptual Model – Service Profile Class

STI · INNSBRUCK

- Non-functional properties
  - **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 (next slide).
  - **serviceCategory -** refers to an entry in some ontology or taxonomy of services. The value of the property is an instance of the class ServiceCategory (next slide).
  - **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
  - **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

NAICS – North American Industry Classification System - http://www.census.gov/naics/
UNSPSC – United Nations Standard Products and Services Code - http://www.unspsc.org/

24

OWL-S
Conceptual Model – Service Profile Class

STI · INNSBRUCK

- Non-functional properties (cont'd)
- **ServiceParameter** class - additional attributes include the quality guarantees that are provided by the service, possible classification of the service, and additional parameters that the service may want to specify
  - **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** class - describes categories of services on the bases of some classification
  - **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

25

OWL-S
Conceptual Model – Service Model Class

STI · INNSBRUCK

- Exposes "how a service works" to enable invocation, composition, monitoring, recovery, etc.
- The Service Model views interaction with the service as a process
  - The Process class represents the central class in the model,
  - Process generates and returns some new information based on the given information and the world state (inputs and outputs - slide # 23).
  - Process produces a change in the world (preconditions and results - slide # 23).
- A Process instance is characterized by the number of properties:
  - Zero or more **inputs** - representing the information that is, under some conditions, required for the performance of the process,
  - Any number of **outputs** - the information that the process provides to the requester,
  - Any number of **preconditions** - which must all hold in order for the process to be successfully invoked,
  - Any number of **effects** - which must all hold after the process invocation.
- The Process class is further specialized into the
  - **Atomic** process - a description of a service that expects one (possibly complex) message and returns one (possibly complex) message.
  - **Simple** process – used for abstraction and simplification purposes.
  - **Composite** process - maintains some state; each message the client sends advances it through the process.

26

OWL-S
Conceptual Model – Service Model Class

STI · INNSBRUCK

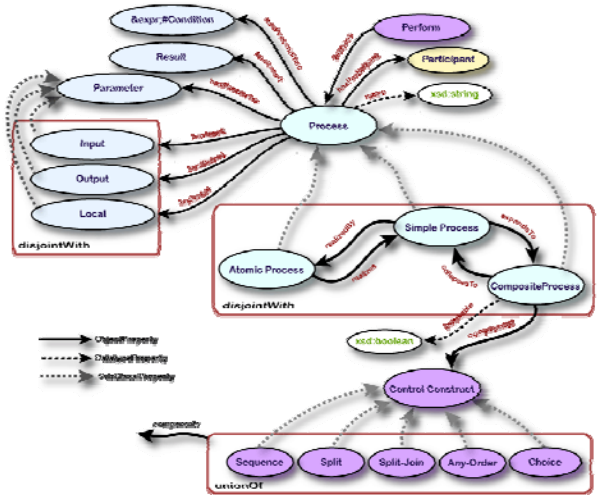


Figure 3 – OWL-S Service Process Model

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

OWL-S
Conceptual Model – Service Model Class – Process Classification

STI · INNSBRUCK

- 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
Conceptual Model – Service Model Class – Complex Process Control Structures

STI · INNSBRUCK

- **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 sub-bag).
- **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

29

OWL-S
Conceptual Model – Service Model Class – Complex Process Control Structures

STI · INNSBRUCK

- **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

30

OWL-S
Conceptual Model – Service Grounding Class

STI · INNSBRUCK

- 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

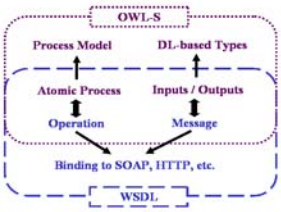


Figure 4 – Mapping between OWL-S and WSDL

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

31

OWL-S
Example

STI · INNSBRUCK

- Bravo Air - fictitious airline site
- The site is providing a service which enables reservation of trips as a composite process consisting of following steps
  - Retrieval of the desired flight details
  - Selection of the available flights, and
  - Booking of the selected flights, which includes following substeps
    - Login and
    - Confirmation of reservation.
- The example consists of the following artifacts
  - BravoAirService.owl - OWL-S service,
  - BravoAirProfile.owl – OWL-S ServiceProfile description,
  - BravoAirProcess.owl – OWL-S ServiceModel description,
  - BravoAirGrounding.owl – OWL-S ServiceGrounding description, and
  - BravoAirGrounding.wsdl – annotated WSDL which points to the specifics OWL-S descriptions used during the processing (not exemplified here).
- Presented examples are lacking full details due to the space constrains.

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

32

OWL-S
Example – Bravo Air Service Ontology

STI · INNSBRUCK

```
<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_ServiceModel"/>
    <!-- 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>
```

33

OWL-S
Example – Bravo Air Service Profile Ontology

STI · INNSBRUCK

```
<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>
```

34

OWL-S
Example – Bravo Air Service Model Ontology

STI · INNSBRUCK

```
<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>
```

35

OWL-S
Example – Bravo Air Service Model Ontology

STI · INNSBRUCK

```
  <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>
```

36

OWL-S
Example – Bravo Air Service Grounding Ontology

STI · INNSBRUCK

```
<rdf:RDF>
 …
  <grounding:WsdlAtomicProcessGrounding rdf:ID="WsdlGrounding_LogIn”>

    <!-- Grounding for the Atomic Process LogIn -->
    <grounding:owlsProcess rdf:resource="&ba_process;#LogIn"/>

    <!-- Reference to the corresponding WSDL operation -->
    <grounding:wsdlOperation rdf:resource="#LogIn_operation"/>

    <!-- Reference to the WSDL input message -->
    <grounding:wsdlInputMessage><xsd:anyURI
     rdf:value="&BravoAirGroundingWSDL;#LogIn_Input"/></grounding:wsdlInputMessage>

    <!-- Mapping of OWL-S inputs to WSDL message parts -->
    <grounding:wsdlInputs rdf:parseType="Collection”>
      <grounding:WsdlInputMessageMap>
        <grounding:owlsParameter rdf:resource="&ba_process;#acctName_In"/>
        <grounding:wsdlMessagePart>
          <xsd:anyURI rdf:value="&BravoAirGroundingWSDL;#acctName"/>
        </grounding:wsdlMessagePart>
      </grounding:WsdlInputMessageMap>
      <grounding:WsdlInputMessageMap>
        <grounding:owlsParameter rdf:resource="&ba_process;#password_In"/>
        <grounding:wsdlMessagePart>
          <xsd:anyURI rdf:value="&BravoAirGroundingWSDL;#password"/>
        </grounding:wsdlMessagePart>
      </grounding:WsdlInputMessageMap>
    </grounding:wsdlInputs>
    <grounding:wsdlReference>
      <xsd:anyURI rdf:value="http://www.w3.org/TR/2001/NOTE-wsdl-20010315"/>
    </grounding:wsdlReference>
  </grounding:WsdlAtomicProcessGrounding>
  …
```

37

OWL-S
Example – Bravo Air Service Grounding Ontology

STI · INNSBRUCK

```
 …
 <grounding:WsdlOperationRef rdf:ID="LogIn_operation”>
    <rdfs:comment>A pointer to the WSDL operation used for SelectAvailableFlight</rdfs:comment>
    <!-- locate port type to be used -->
    <grounding:portType>
      <xsd:anyURI rdf:value="&BravoAirGroundingWSDL;#LogIn_PortType"/>
    </grounding:portType>
    <!-- locate operation to be used -->
    <grounding:operation>
      <xsd:anyURI rdf:value="&BravoAirGroundingWSDL;#LogIn_operation"/>
    </grounding:operation>
  </grounding:WsdlOperationRef>
…
</rdf:RDF>
```

38

OWL-S
Tool support

STI · INNSBRUCK

- 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

39

OWL-S
Tool support

STI · INNSBRUCK

- 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.

40

OWL-S
Relationship to WSMO

STI · INNSBRUCK

- 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

41

OWL-S
Criteria discussion

STI · INNSBRUCK

- 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
  - Does not 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
  - Does not have it
- Service vs. Web services

42

STI · INNSBRUCK

Technical solution
METEOR - S

METEOR-S
Overall approach

STI · INNSBRUCK

- METEOR-S[3] 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.
- As opposed to the top-down approach embraced by WSMO and OWL-S, the project is promoting bottom-up approach.
- Conceptual model is simple and quite generic
  - Relying on Semantically Annotated WSDL (explained in next slides).
- Based on this simple model the project is providing a set of tools
  - Web Service Annotation Framework,
  - Web Service Discovery Infrastructure,
  - Web Service Composition Framework.

[3] http://lsdis.cs.uga.edu/projects/meteor-s

METEOR-S
Conceptual Model

STI · INNSBRUCK

- The central point of METEOR-S approach is Semantically Annotated WSDL (SAWSDL)
- It is a way to add semantic annotations to various parts of a WSDL document
  - Input and output message structures, interfaces and operations.
- Relying to simple extension attributes
  - Compliant to WSDL v2.0 and v1.1, and XML Schema
- Annotations can be used for various purposes:
  - WSDL interfaces and operations with categorization information used to advertise Web services
  - XML Schema types to foster discovery and composition
  - Specifying the data mapping of XML Schema to/from an ontology used during invocation (possible mediation)
- Independent on the ontology expression language and mapping languages

45

METEOR-S
Conceptual Model

STI · INNSBRUCK

- Extension attributes of SAWSDL are:
  - *modelReference*
    - Pointers to a concept in some semantic model.
    - Annotates XML Schema type definitions, element declarations, and attribute declarations, WSDL interfaces, operations, and faults.
  - *liftingSchemaMapping*
    - Added to XML Schema element declarations and type definitions for specifying lifting mappings between semantic data and XML.
  - *loweringSchemaMapping*
    - Added to XML Schema element declarations and type definitions for specifying lowering mappings between semantic data and XML.

46

METEOR-S
Example

STI · INNSBRUCK

```
<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>
```

47

METEOR-S
Tools

STI · INNSBRUCK

- Web Service Annotation Framework for semi-automatic 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.
- The framework 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.

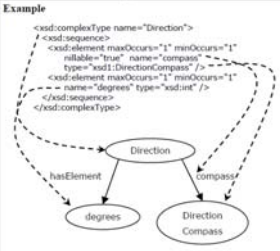
48

METEOR-S
Tools – Annotation Framework - Example

STI · INNSBRUCK

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

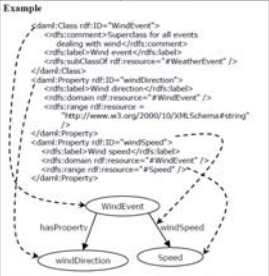
Example



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

49

METEOR-S
Tools

STI · INNSBRUCK

- Web Service Discovery Infrastructure
  - Attempts to enhance Web Service discovery infrastructure by using semantics,
  - Relies strongly on the existing UDDI Web Service registries,
  - Unifies access to the Web Service registries and ontology registries over P2P infrastructure.
  - User is expressing his requirements through the Service Templates which specify the inputs and outputs using concepts from the registered domain specific ontologies.
  - Semantic matchmaking is performed between the SAWSDL-enriched service descriptions and the Service Templates.
- Web Service Composition Framework
  - 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.

50

METEOR-S
Tools – Composition Framework - Example

STI · INNSBRUCK

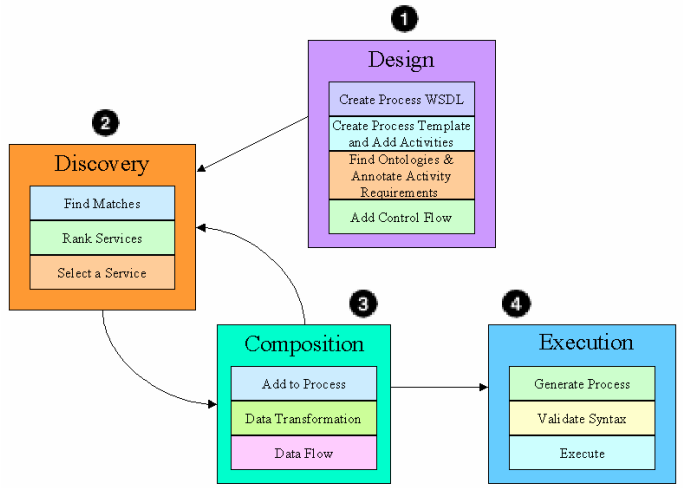


Figure taken from Sivashanmugam, K., et al. Framework for Semantic Web Process Composition, Special Issue of the International Journal of Electronic Commerce (IJEC), Eds: Christoph Bussler, Dieter Fensel, Norman Sadeh, Feb 2004

51

METEOR-S
Relationship to WSMO

STI · INNSBRUCK

- With its minimalistic approach can be viewed as orthogonal to WSMO,
- It can reference WSMO descriptions from its annotations,
- All the SAWSDL referenced concepts can be annotated in WSML/WSMO.
- SAWSDL 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).
- SAWSDL has a strong industrial point
  - WSMO/L/X working groups are looking at closer alignment with industry standards and adoption of SAWSDL as a possible future grounding formalism for WSMO.

52

METEOR-S
Criteria discussion

STI · INNSBRUCK

- 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

53

STI · INNSBRUCK

Technical solution
Semantic Web Services Framework (SWSF)

54

SWSF
Overall Approach

STI · INNSBRUCK

- Semantic Web Services Framework (SWSF) has its roots in OWL-S and the Process Specification Language (PSL).
- It is 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.**
- It is a complex solution which has not received much of the attention in the community
- General lack of tool support
- Comprehensive examples are missing

55

SWSF
Conceptual Model

STI · INNSBRUCK

- 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
  - First-order Logic Ontology for Web Services (FLOWS), and
    - Relies on the semantic of SWLS-FOL
  - Rule Ontology for Web Services (ROWS)
    - Relies on the semantics of SWLS-Rules.
- In order to describe behavior of a service based on PSL approach two fundamental elements are added:
  - Structured notion of atomic processes, and
  - Infrastructure for specifying various forms of data flow.

56

SWSF
Relationship to WSMO

STI · INNSBRUCK

- 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 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.

57

SWSF
Criteria discussion

STI · INNSBRUCK

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

58

Technical solution
Internet Reasoning Service - III

Internet Reasoning Service (IRS-III)
Overall Approach

- 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 (IRS-II) by incorporating and extending the WSMO ontology within the IRS-III server, browser and API.

Internet Reasoning Service (IRS-III)
Principal Characteristics

STI · INNSBRUCK

- Ontological separation of User and Web Service Contexts
  - Consumer and service exist in their own contexts which should be modeled with distinct ontological structures.
- Capability Based Invocation
  - Users can invoke a Web Service by specifying a concrete desired capability. Semantic descriptions in IRS-III are operational.
- Ease of Use
  - Interfaces are designed to hide the complexities surrounding the creation of SWS-based applications.
- One Click Publishing
  - Stand alone code written in a standard programming language (Java or LISP) can be turned into a Web Service by only one-click.
- Agnostic to Service Implementation Platform
  - IRS-III is not making strict assumptions about the underlying service implementation platform.
  - It accepts the current dominance of Web Service stack of standards.

61

Internet Reasoning Service (IRS-III)
Principal Characteristics

STI · INNSBRUCK

- Connected to External Environment
  - Functions and relations can make extra logical calls (e.g. invoking a Web Service) which results can be integrated smoothly with the internal reasoning.
- Open
  - IRS-III clients are based on publicly accessible Java APIs, components of the IRS-III server are Semantic Web Services which fosters flexibility.
- Complete Descriptions
  - Data which can be represented is represented. It is not possible a priori to know which specific data will be required for SWS related reasoning.
- Inspectable
  - Making the semantic descriptions accessible in a human readable form. The key is that the content and form are easily understandable by SWS application builders.
- Interoperable with SWS Frameworks and Platforms
  - IRS-III has an OWL-S import mechanism. It is interoperable with the WSMO reference implementation WSMX.

62

Internet Reasoning Service (IRS-III)
Features

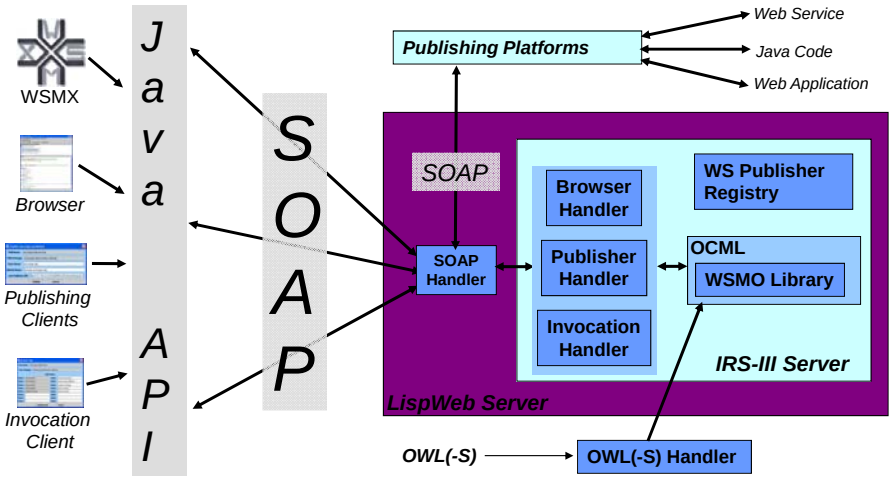
STI · INNSBRUCK

- 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

63

Internet Reasoning Service (IRS-III)
Architecture

STI · INNSBRUCK



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

64

Internet Reasoning Service (IRS-III)
Comparison with WSMX

STI · INNSBRUCK

- Underlying language
  - In WSMX underlying language is WSML, while in IRS-III it is OCML.
- Goals
  - In IRS-III goals have inputs and outputs while in WSMX they have pre- and post-conditions.
- Discovery mechanism
  - IRS-III broker is finding applicable Web Services via mediators
    - Used mediator within WS capability
    - Mediator source = goal
- WSMX does not provide One-click-publishing

65

Summary

STI · INNSBRUCK

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

66

References

STI · INNSBRUCK

- Mandatory reading:
  - Dieter Fensel, Mick Kerrigan, Michal Zaremba (Eds.), Implementing *Semantic Web Services: The SESA Framework*. Springer-Verlag, 2008. (Chapter 13)
- Further reading:
  - 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
  - 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
  - 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
  - Abhijit Patil, Swapna Oundhakar, Amit Sheth, Kunal Verma, METEOR-S Web service Annotation Framework, Proceeding of the World Wide Web Conference, July 2004 (Proceeding of the World Wide Web Conference, July 2004) http://lsdis.cs.uga.edu/lib/download/POSV-WWW2004.pdf
  - Verma, K., Sivashanmugam, K. , Sheth, A., Patil, A., Oundhakar, S. and Miller, J. METEOR–S WSDI: A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web Services , Journal of Information Technology and Management, 2004. http://lsdis.cs.uga.edu/lib/download/VSS+03-TM06-003-METEOR-S-WSDI.pdf

67

References

STI · INNSBRUCK

- Wikipedia and links
  - Semantic Web Services http://en.wikipedia.org/wiki/Semantic_Web_Services
  - OWL-S http://en.wikipedia.org/wiki/OWL-S
  - WSDL-S http://en.wikipedia.org/wiki/Web_Services_Semantics

68

Next Lecture

STI · INNSBRUCK

#	Title
1	Introduction
2	Web Science + TourPack project (separate slideset)
3	Service Science
4	Web services
5	Web2.0 services + ONLIM APIs (separate slideset)
6	Semantic Web
7	Semantic Web Service Stack (WSMO, WSML, WSMX)
8	OWL-S and the others
9	**Semantic Services as a Part of the Future Internet and Big Data Technology**
10	Lightweight Annotations
11	Linked Services
12	Applications
13	Mobile Services

69

Questions?

STI · INNSBRUCK

70