STI · INNSBRUCK

Semantic Web Services

Web Service Execution Environment (WSMX)

Lecture VIII – 14th May 2009

Srdjan Komazec

Where are we?

STI · INNSBRUCK

#	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

Outline

STI · INNSBRUCK

- Motivation
  - Semantically Enabled Service-oriented Architecture (SESA)
- Technical solution
  - Introduction
    - Relation to the WSMO and WSML
    - Design principles
    - Lifecycle
  - Web Service Execution Environment (WSMX)
    - Architecture
    - Components
    - System entry points
    - Execution semantics
- Illustration by larger example

STI · INNSBRUCK

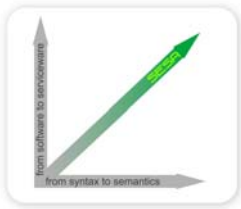
Motivation

Motivation
Semantically Enabled Service-oriented Architecture

- Service Orientation
  - Service-Oriented Computing (SOC)
    - Services as the fundamental elements for the development of rapid, low-cost, and easily integrable enterprise applications,
  - Service-Oriented Architecture (SOA)
    - SOC can be abstractly implemented by SOA,
    - From ***function*** to ***object*** to ***service***,
    - SOA requirements: loose coupling, implementation neutrality, flexible configuration, long lifetime, granularity and teams.
- Existing technologies and SOA solutions are…
  - … difficult to scale without a proper degree of automation,
  - … partial solution to interoperability.
- SESA represents SOA empowered by adding semantics as a means to deal with heterogeneity and mechanization of service usage.

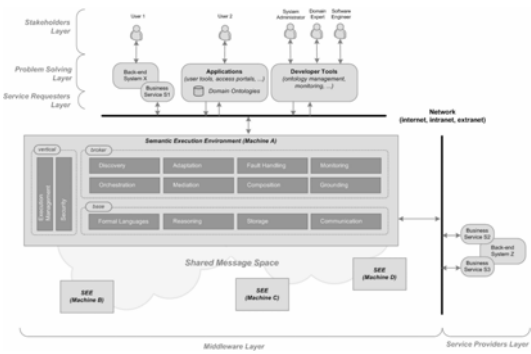
Motivation
Semantically Enabled Service-oriented Architecture

- Application of SESA offers a scalable integration, more adaptive to changes
  - service offerings and required capabilities are described by semantically rich and formal service models,
  - exchanged data is also semantically described, and
  - reasoning provides total or partial automation of tasks.
- A SESA implementation should build a layer on top of the existing technologies (e.g. Web Services).



WSMX is an implementation of SESA.

Motivation
SESA Architecture



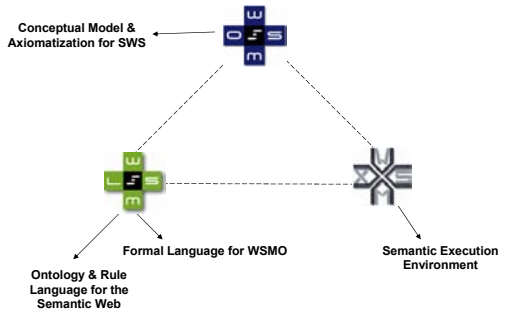
Fensel, D.; Kerrigan, M.; Zaremba, M. (Eds): *Implementing Semantic Web Services: The SESA Framework. Springer 2008.*

Motivation

- Middleware for Semantic Web Services
  - Allows service providers to focus on their business,
- Reference implementation for WSMO
  - Eat our own cake,
- Environment for goal based discovery and invocation
  - Run-time binding of service requesters and providers,
- Provide a flexible Service Oriented Architecture
  - Add, update, remove components at run-time as needed,
- Keep open source to encourage participation
  - Developers are free to use in their own code, and
- Define formal execution semantics
  - Unambiguous model of system behavior.

Technical solution

Introduction
Relation to WSMO and WSML



Introduction
WSMX...

- … is comprehensive software framework for runtime binding of service requesters and service providers,
- … interprets service requester's goal to
  - discover matching services,
  - select (if desired) the service that best fits,
  - provide data/process mediation (if required), and
  - make the service invocation,
- … is reference implementation for WSMO,
- … has a formal execution semantics, and
- … is service oriented, event based and has pluggable architecture
  - Open source implementation available through Source Forge,
  - based on microkernel design using technologies such as JMX.

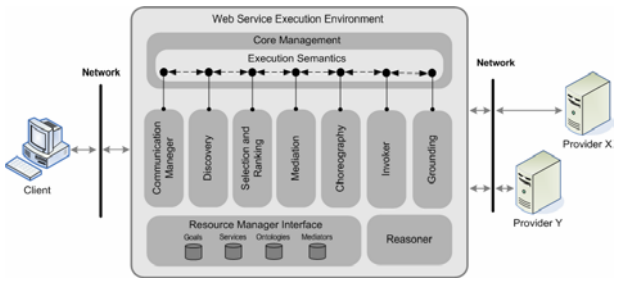
Introduction
Design principles

- Service-oriented principle
  - Service reusability, loose coupling, abstraction, composability, autonomy, discoverability,
- Semantic Principle
  - Rich and formal description of information and behavioral models enabling automation of certain tasks by means of logical reasoning,
- Problem-solving principle
  - Goal-based discovery and invocation of services, and
- Distributed principle
  - Executing process across a number of components/services over the network, thus promoting scalability and quality of process.

Lifecycle

1. Discovery - determines usable services for a request,
2. Composition - combine services to achieve a goal,
3. Selection - chooses most appropriate service among the available ones,
4. Mediation- solves mismatches (data, protocol, process) hampering interoperation,
5. Choreography – interactions and processes between the service providers and clients,
6. Grounding – lifting and lowering between the semantic and syntactic data representations, and
7. Invocation - invokes Web service following programmatic conventions.

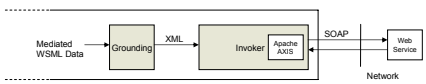
WSMX
Current middleware status



WSMX Components

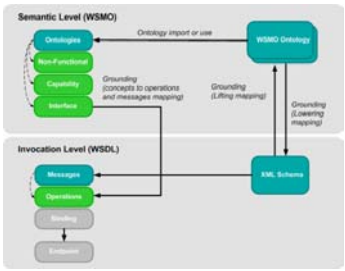
WSMX Components
Communication Manager, Invoker and Grounding

- Responsible for interaction with services and entities that are external to WSMX.
- Should be open to support as many transport and messaging protocols as possible (transparently to WSMX).
- WSMX uses
  - The SOAP implementation from Apache AXIS, and
  - The Apache Web Service Invocation Framework (WSIF) .
- Both RPC and Document style invocations possible



WSMX Components
Grounding

- WSMO service descriptions are grounded to WSDL by the means of XSLT lifting and lowering



Jacek Kopecký et al. D24.2v0.1. WSMO Grounding, WSMO Working Draft 27 April 2007. http://wsmo.org/TR/d24/d24.2/v0.1

WSMX Components
Grounding - Example

```
<xsl:stylesheet version="2.0"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:p="http://www.wsmo.org/sws-challenge/ShipmentOntologyProcess#"
  xmlns:muller="http://www.example.org/muller/"
  exclude-result-prefixes="#all"
  xmlns:helper="java:ie.deri.wsmx.commons.Helper">

<xsl:output method="xml" omit-xml-declaration="yes"/>

<xsl:template match="/muller:invokePriceResponse">

<xsl:variable name="randomId" select="helper:getRandomId()"/>

 <rdf:RDF>
  <rdf:Description about="http://example.com/testresponse{$randomId}">
   <rdf:type rdf:resource="http://www.wsmo.org/sws-challenge/ShipmentOntologyProcess#PriceQuoteResp"/>
   <p:price rdf:datatype="http://www.w3.org/2001/XMLSchema#decimal"><xsl:value-of
    select="number(./muller:price)"/></p:price>
  </rdf:Description>
 </rdf:RDF>
</xsl:template>
</xsl:stylesheet>
```

An example of lifting XML data to RDF

WSMX Components
Discovery

- Responsible for finding appropriate Web Services capable of fulfilling a goal
- Different techniques available
  - trade-off: ease-of-provision vs. accuracy
  - resource descriptions & matchmaking algorithms

Key Word Matching
- match natural language key words in resource descriptions,

Controlled Vocabulary
- ontology-based key word matching, and

Semantic Matchmaking
- what Semantic Web Services aim at.

Ease of provision

Possible Accuracy

WSMX Components
Discovery – Key Word Matching

- Allows for a fast filtering and ranking of the huge number of available services rather quickly.
- Nonfunctional properties from the Dublin Core namespace (e.g. dc#description) are candidates for indexing and querying.
- Dictionaries of synonyms (WordNet) can be used to discover more services.

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-rule"

namespace {_"http://www.wsmo.org/sws-challenge/WSMuller#",
        dc _"http://purl.org/dc/elements/1.1#"}

webService WSMuller
   nfp
      dc#title hasValue "Muller Web Service"
      dc#description hasValue "We ship to Africa, North America, Europe, Asia (all countries)."
      dc#contributor hasValue "Maciej Zaremba, Matt Moran, Tomas Vitvar, Thomas Haselwanter"
   endnfp

capability WSMullerCapability
...
```

WSMX Components
Discovery – Simple Semantic Description

= G = WS

Exact Match:
G, WS, O, M ⊨ ∀x. (G(x) <=> WS(x))

PlugIn Match:
G, WS, O, M ⊨ ∀x. (G(x) => WS(x))

Subsumption Match:
G, WS, O, M ⊨ ∀x. (G(x) <= WS(x))

Intersection Match:
G, WS, O, M ⊨ ∃x. (G(x) ∧ WS(x))

Non Match:
G, WS, O, M ⊨ ¬∃x. (G(x) ∧ WS(x))

Keller, U.; Lara, R.; Polleres, A. (Eds): *WSMO Web Service Discovery*. WSML Working Draft D5.1, 12 Nov 2004.

WSMX Components
Discovery – Simple Semantic Description - Example

$$Trip \sqsubseteq (=1 startLocation)$$
$$Trip \sqsubseteq (=1 endLocation)$$
$$NightTrip \sqsubseteq Trip$$
$$City \sqsubseteq Location$$
$$EuropeanCity \sqsubseteq City$$
$$GermanCity \sqsubseteq EuropeanCity$$
$$AustrianCity \sqsubseteq EuropeanCity$$
$$GermanCity \sqcap AustrianCity \sqsubseteq \bot$$
$$Trip \sqcap Location \sqsubseteq \bot$$
$$\{Salzburg\} \sqcap \{Innsbruck\} \sqsubseteq \bot$$
$$\{Hamburg\} \sqcap \{Frankfurt\} \sqsubseteq \bot$$

GermanCity(Hamburg), GermanCity(Frankfurt)
AustrianCity(Innsbruck), AustrianCity(Salzburg)

Domain knowledge

$$Goal\_TripAT2DE \equiv \exists hasPostCondition.(Trip \sqcap \exists start.AustrianCity \sqcap \exists end.GermanCity)$$
$$Goal\_TripAustria \equiv \exists hasPostCondition.(Trip \sqcap \exists start.AustrianCity \sqcap \exists end.AustrianCity)$$
$$Goal\_TripEurope \equiv \exists hasPostCondition.(Trip \sqcap \exists start.EuropeanCity \sqcap \exists end.EuropeanCity)$$

Generic goals

$$Goal\_TripIBK2FRA \equiv \exists hasPostCondition.(Trip \sqcap \exists start.\{Innsbruck\} \sqcap \exists end.\{Frankfurt\})$$
$$Goal\_TripIBK2SZG \equiv \exists hasPostCondition.(Trip \sqcap \exists start.\{Innsbruck\} \sqcap \exists end.\{Salzburg\})$$

Specific goals

$$Service\_VTA \equiv \exists hasPostCondition.(Trip \sqcap \exists start.(GermanCity \sqcup AustrianCity) \sqcap \exists end.(GermanCity \sqcup AustrianCity))$$
$$Service\_DB \equiv \exists hasPostCondition.(Trip \sqcap \exists start.GermanCity \sqcap \exists end.GermanCity)$$
$$Service\_OEBB \equiv \exists hasPostCondition.(Trip \sqcap \exists start.AustrianCity \sqcap \exists end.AustrianCity$$
$$Service\_DBNight \equiv \exists hasPostCondition.(NightTrip \sqcap \exists start.\{Innsbruck \sqcup Salzburg\} \sqcap \exists end.GermanCity))$$

Web services

Lara, R., Lausen, H (Eds): *WSMO Discovery Engine*. WSML Working Draft D5.2, 26 Nov 2004.

WSMX Components
Discovery – Simple Semantic Description - Example

- Exact match
- Plug in match
- Subsumption match
- Intersection match

$$(1) Goal\_TripAustria \equiv Service\_OEBB$$
$$(2) Goal\_TripAT2DE \sqsubseteq Service\_VTA$$
$$(3) Goal\_TripIBK2FRA \sqsubseteq Service\_VTA$$
$$(4) Goal\_TripAustria \sqsubseteq Service\_VTA$$
$$(5) Goal\_TripIBK2SZG \sqsubseteq Service\_VTA$$
$$(6) Goal\_TripIBK2SZG \sqsubseteq Service\_OEBB$$
$$(7) Service\_VTA \sqsubseteq Goal\_TripEurope$$
$$(8) Service\_DB \sqsubseteq Goal\_TripEurope$$
$$(9) Service\_OEBB \sqsubseteq Goal\_TripEurope$$
$$(10) Service\_DBNight) \sqsubseteq Goal\_TripEurope$$
$$(11) \neg(Goal\_TripIBK2FRA \sqcap Service\_DBNight \sqsubseteq \bot)$$

WSMX Components
Ranking and Selection

- One service which best satisfies the user preferences is selected from the candidate services returned by the service discovery.
- Selection
  - determines best candidate out of discovered WS,
- Ranking
  - determines a priority list of discovered WS.
- The process is run after “functional” discovery
- Criteria:
  - Quality of Service (security, robustness, availability),
  - Context (regional, business / social communities),
  - Preferences and policies,
  - Financial criteria,
  - …

WSMX Components
Ranking and Selection

- Ontologies for specifying QoS. http://www.wsmo.org/ontologies/nfp/
  - set of 17 ontologies (i.e. locative, temporal, availability, price, trust, security, etc.)
  - provide the terminology needed to specify QoS aspects of services

WSMX Components
Attaching non-functional properties descriptions to services

- Model

```
Class nonFunctionalProperty
      hasAnnotations type annotations
      hasDefinition type logicalExpression
```

- Example

```
webService _"http://example.org/ws"
importsOntology {_"http://www.wsmo.org/ontologies/nfp/priceNFPOntology",
                 _"http://www.example.org/ontologies/humansOntology",
                 _"http://www.example.org/ontologies/clientsOntology"}

nonFunctionalProperty

   price hasValue ?price
     definedBy ?client[age hasValue ?age] memberOf hu#human and ?age[amount hasValue ?years,
               units hasValue hua#YearsDuration] memberOf hua#age and (greaterEqual(?years, 60) or
               lessEqual(?years,10)) implies ?price[hasAmount hasValue ?amount, hasCurrency
               hasValue cur#Euro] memberOf poa#AbsolutePrice and lessEqual(?amount, 10).
```

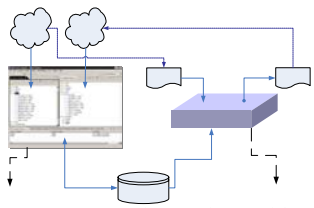

WSMX Components
Ranking and Selection - Example

- Extension of SWS Challenge Shipment Scenario
- *Discounts* and *obligations* modeled for all services in the scenario

```
axiom DefinitionDiscounts
  definedBy
        //delivery price per order/package depends on the weight
        //of the package: Racer charges 20 cents per kilo
        computeDeliveryPrice(?order, ?deliveryPrice):-
         ?order[so#paymentMethod hasValue ?paymentMethod, so#package hasValue ?package] and
         ?package[so#declaredValue hasValue ?value, so#weight hasValue ?weight] and
         ?deliveryPrice = (?weight * 0.2).

       //6% discount if payed cash and base delivery price more than or equal 100$
       computeDeliveryReducedPriceCash(?order, ?reducedPrice):-
        computeDeliveryPrice(?order, ?price) and ?order[so#paymentMethod hasValue ?paymentMethod] and
        ?paymentMethod = pay#CashInstrument and ?price >= 100 and
        ?reducedPrice = (?price * 0.94).

       computeDeliveryReducedPriceCash(?order, ?reducedPrice):-
        computeDeliveryPrice(?order, ?price) and ?order[so#paymentMethod hasValue ?paymentMethod] and
        (?paymentMethod != pay#CashInstrument or ?price < 100) and
        ?reducedPrice = ?price.

       discounts(?order,?reducedPrice):-
         computeDeliveryPrice(?order,?price1) and
         computeDeliveryReducedPriceCash(?order,?price2) and
         list#min(?price1,?price2,?reducedPrice).
```


WSMX Components
Data Mediation

- Ontology to ontology mediation
- A set of mapping rules are defined and then executed
  - Ontology Mapping Language
- Initially rules are defined semi- automatic
- Create for each source instance the target instance(s)



Fensel, D.; Kerrigan, M.; Zaremba, M. (Eds): *Implementing Semantic Web Services: The SESA Framework. Springer 2008.*

WSMX Components
Data Mediation

Design-time
- Inputs
  - Source Ontology and Target Ontology
- Features
  - Graphical interface
  - Set of mechanism towards semi-automatic creation of mappings
  - Capturing the semantic relationships identified in the process
  - Storing these mappings in a persistent storage
- Output
  - Abstract representation of the mappings

Run-time
- Main Mediation Scenario: Instance Transformation
- Inputs
  - Incoming data
    - Source ontology instances
- Features
  - Completely automatic process
  - Grounding of the abstract mappings to a concrete language
    - WSML
  - Uses reasoner to evaluate the mapping rules
- Outputs
  - Mediated data
    - Target ontology instances

WSMX Components
Data Mediation - Example

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-flight"
namespace { _"http://deri.org/iswc2005tutorial/ontologies/travel1#"}

ontology travel1

concept ticket
    type ofType _string
    departure_city ofType _string
    departure_code ofType _string
    arrival_city ofType _string
    arrival_code ofType _string
    departure_date ofType date
    arrival_date ofType date
    departure_time ofType time
    arrival_time ofType time
    issuing_terms ofType terms
    firstName ofType _string
    lastName ofType _string
...
```

Source ontology

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-
        flight"
namespace {
        _"http://deri.org/iswc2005tutorial/ontologies/travel2#"}

ontology travel2

concept travelVoucher
    type ofType _string
    bearer ofType name
    toFrom ofType tripPoints
    departureDate ofType date
    arrivalDate ofType date
    departureTime ofType time
    arrivalTime ofType time
    terms ofType payment
    deliveryDate ofType date
...
```

Destination ontology

WSMX Components
Data Mediation - Example

```
<Alignment>
 <dc:identifier rdf:resource="m2"/>
 <onto1>
   <formalism name="WSML"
      uri="http://www.wsmo.org/wsml"/><uri>http://deri.org/iswc2005tutorial/ontologies/travel1#travel1</uri>
 </onto1>
 <onto2>
   <formalism name="WSML"
      uri="http://www.wsmo.org/wsml"/><uri>http://deri.org/iswc2005tutorial/ontologies/travel2#travel2</uri>
 </onto2>
 <map>
   …
   <Cell
      id="http://deri.org/iswc2005tutorial/ontologies/travel1#tickethttp://deri.org/iswc2005tutorial/ontologies/travel2#travelVoucher">
     <entity1>
       <Class rdf:about="http://deri.org/iswc2005tutorial/ontologies/travel1#ticket"></Class>
     </entity1>
     <entity2>
       <Class rdf:about="http://deri.org/iswc2005tutorial/ontologies/travel2#travelVoucher"></Class>
     </entity2>
     <measure>1.0</measure>
     <relation>ClassMapping</relation>
   </Cell>
   …
 </map>
</Alignment>
```

Mapping between two concepts

WSMX Components
Data Mediation - Example

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-
        flight"
namespace {
        _"http://deri.org/iswc2005tutorial/ontologies/travel1#"}

ontology travel1

instance my_ticket_input memberOf ticket
    type hasValue "flight"
    firstName hasValue "Adrian"
    lastName hasValue "Mocan"
    arrival_date hasValue my_arrival_date
    departure_date hasValue my_departure_date
    arrival_time hasValue my_arrival_time
    departure_time hasValue my_departure_time
    departure_city hasValue "Innsbruck"
    departure_code hasValue "INN"
    arrival_city hasValue "Rome"
    arrival_code hasValue "RO"
    issuing_terms hasValue my_terms
```

Source instances

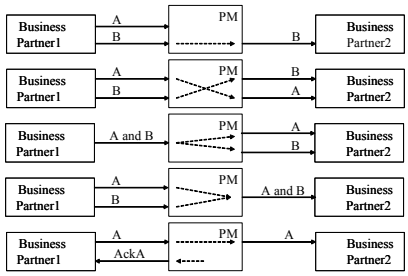
```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-
        flight"
namespace {
        _"http://deri.org/iswc2005tutorial/ontologies/travel2#"}

ontology travel2

instance expected_travelVoucher memberOf travelVoucher
        departureDate hasValue expected_departureDate
        terms hasValue expected_payment
        arrivalDate hasValue expected_arrivalDate
        bearer hasValue expected_name
        departureTime hasValue expected_departureTime
        arrivalTime hasValue expected_arrivalTime
        type hasValue "flight"
```

Destination instances

WSMX Components
Process Mediation

- Requester and provider have their own communication patterns
- Only if the two match precisely, a direct communication may take place
- At design time equivalences between the choreographies' conceptual descriptions is determined and stored as set of rules
- The Process Mediator provides the means for runtime analyses of two choreography instances and uses mediators to compensate possible mismatches

WSMX Components
Process Mediation



Fensel, D.; Kerrigan, M.; Zaremba, M. (Eds): *Implementing Semantic Web Services: The SESA Framework. Springer 2008.*

WSMX Components
Process Mediation - Example

- Not a priori compatible behavior interfaces for communication & information interchange
- Partially resolvable by "process mediation patterns"

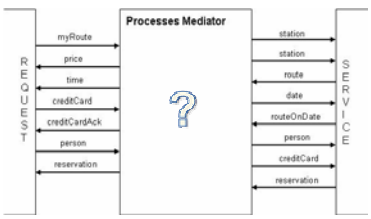
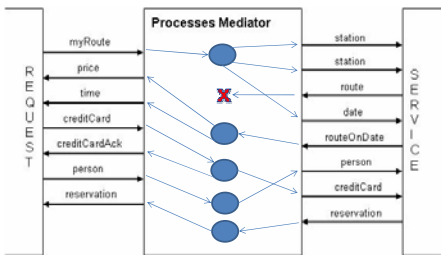


Figure taken from Emilia Cimpian, D13.7 v0.1 Process Mediation in WSMX, WSMX Working Draft 08 July 2005

WSMX Components
Process Mediation - Example



WSMX Components
Choreography

- Requester and provider have their own observable communication patterns
  - Choreography part of WSMO
- Choreography instances are loaded for the requester and provider
  - Both requester and provider have their own WSMO descriptions
- Abstract State Machines (ASM)-based Choreography Engine
  - Evaluation of transition rules
    - prepares the available data
  - Sends data to the Process Mediator
    - filters, changes or replaces data
  - Receives data from PM and forwards it to the Communication manager
    - data to be finally sent to the communication partner

WSMX Components
Choreography - Example

```
choreography WSMullerShipmentOrderChoreography
   stateSignature WSMullerShipmentOrderStateSignature
   …
   in sop#ShipmentOrderReq withGrounding { _"http://sws-
      challenge.org/shipper/v2/muller.wsdl#wsdl.interfaceMessageReference(muller/ShipmentOrder/in0)"}
   in so#ContactInfo
   in so#ShipmentDate
   in so#Package
   in so#Address
   out sop#ShipmentOrderResp

   transitionRules WSMullerShipmentOrderTransitionRules
   forall {?request} with
      (?request memberOf sop#ShipmentOrderReq)
   do
      add(_#1 memberOf sop#ShipmentOrderResp)
      delete(?request memberOf sop#ShipmentOrderReq)
   endForall
```

```
<shipmentOrderReq(soi#MoonContactInfo, soi#shipmentDate1, package, soi#SzyslakContactInfo),
 package(1, 7.0, 6.0, 4.0, 1.0),
 shipmentDate1("2009-01-21T13:00:00.046Z", "2009-01-22T13:00:00.046Z")>
```

S1

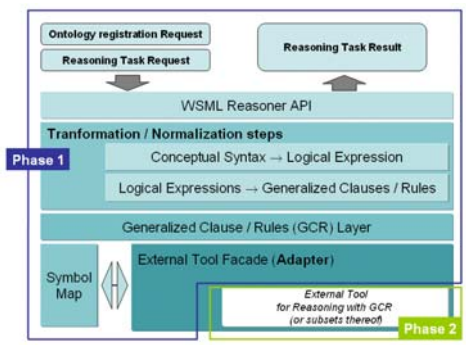
```
<shipmentOrderResp("2009-01-21T15:00:00.046Z", 65.03),
 package(1, 7.0, 6.0, 4.0, 1.0),
 shipmentDate1("2009-01-21T13:00:00.046Z", "2009-01-22T13:00:00.046Z")>
```

S2

WSMX Components
Resource Manager

- Stores internal memory model to a data store
- Decouples storage mechanism from the rest of WSMX
- Data model is compliant to WSMO API
- Independent of any specific data store implementation i.e. database and storage mechanism
- Maintains six repositories to store
  - WSMO top level entities, i.e.
    - Goals,
    - Web Service descriptions,
    - Mediators, and
    - Ontologies.
  - Event data and intermediate messages
  - WSDL descriptions

WSMX Components
A Generic Framework for Reasoning with WSML



Fensel, D.; Kerrigan, M.; Zaremba, M. (Eds): *Implementing Semantic Web Services: The SESA Framework. Springer 2008.*

WSMX Entry Points

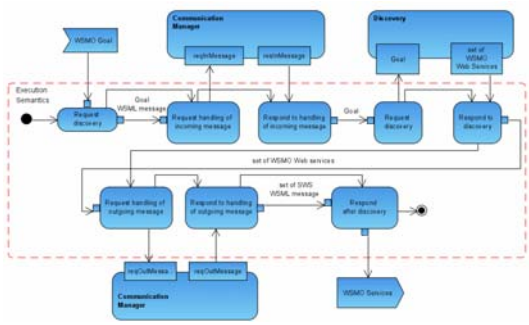
- Represent input ports to which messages can be sent for initiating specific execution semantics.
  - getWebServices(WSMLDocument): Web Services
    - A service requester wishes to discover a list of SWS fulfilling its requirements provided by as a goal description using WSML.
    - A set of WSML Web Service descriptions whose capability matches the goal is returned.
  - invokeWebService(WSMLDocument, Context): Context
    - Used to invoke already known Semantic Web Service by relying on data provided in the form of WSML ontology and conversation context.
  - achieveGoal(WSMLDocument): Context
    - A service requester wishes to use WSMX for all aspects of goal-based service invocation (discovery, mediation, invocation) by providing both goal and data in the single WSML document.
    - Processing of the message is identified by the conversation context .

Execution Semantics

Execution Semantics

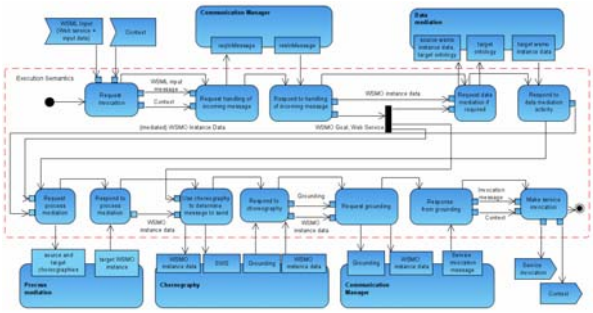
- Formal description of the operational behavior of the system in terms of computational steps
  - Greater flexibility in SESA implementations,
  - Foundations for model testing,
  - Executable representation, and
  - Improved model understanding among humans.
- Mandatory execution semantics
  - Goal-Based Web Service Discovery
  - Web Service Invocation
  - Goal-Based Service Execution

Execution Semantics
Goal-Based Web Service Discovery



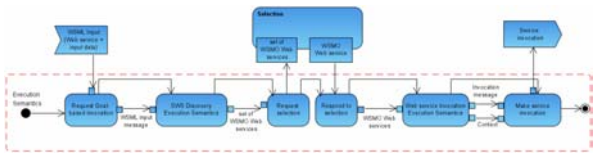
Fensel, D.; Kerrigan, M.; Zaremba, M. (Eds): *Implementing Semantic Web Services: The SESA Framework. Springer 2008.*

Execution Semantics
Web Service Invocation



Fensel, D.; Kerrigan, M.; Zaremba, M. (Eds): *Implementing Semantic Web Services: The SESA Framework. Springer 2008.*

Execution Semantics
Goal-Based Service Execution



Fensel, D.; Kerrigan, M.; Zaremba, M. (Eds): *Implementing Semantic Web Services: The SESA Framework. Springer 2008.*

Illustration by larger example

Illustration by larger example
Scenario description

- The goal is to discover a suitable solution for the transportation of a package with defined size and weight
- Candidate Web Services have different constraints regarding the transportation destinations, package size and weight acceptance, as well as pricing schemas
- For more information visit:
  - http://sws-challenge.org/wiki/index.php/Scenario:_Shipment_Discovery

Illustration by larger example
Goal description

I want to have my package shipped from CA, USA to Tunis, Africa size (7/6/4), weight 1 lbs, the cheaper the better.

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-flight"

goal GoalA1

capability GoalA1Capability
  postcondition
    definedBy
      ( ?x[sop#price hasValue ?price] memberOf
sop#PriceQuoteResp
        and sop#isShipped(shipmentOrderReq) ).

interface GoalA1Interface
  choreography GoalA1Choreography
    stateSignature GoalA1StateSignature

    in sop#ShipmentOrderReq
    out sop#ShipmentOrderResp

    transitionRules GoalA1TransitionRules

      forall {?request} with
        (?request memberOf sop#ShipmentOrderReq)
      do
        add(_#1 memberOf sop#ShipmentOrderResp)
      endForall
```

```
ontology GoalRequest

instance shipmentOrderReq memberOf sop#ShipmentOrderReq
  sop#from hasValue soi#MoonContactInfo
  sop#shipmentDate hasValue soi#shipmentDate1
  sop#package hasValue package
  sop#to hasValue soi#SzyslakContactInfo

instance package memberOf soi#Package
  soi#quantity hasValue 1
  soi#length hasValue 7.0
  soi#width hasValue 6.0
  soi#height hasValue 4.0
  soi#weight hasValue 1.0

instance shipmentDate1 memberOf soi#ShipmentDate
  soi#earliest hasValue "2009-01-21T13:00:00.046Z"
  soi#latest hasValue "2009-01-22T13:00:00.046Z"
```


Illustration by larger example
AchieveGoal execution semantics

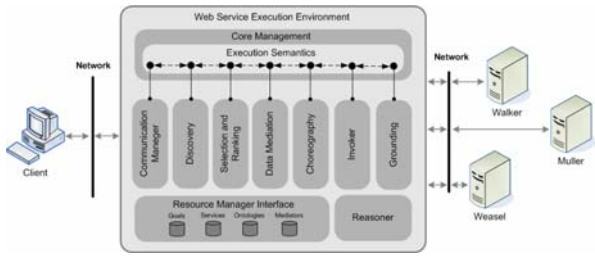


Illustration by larger example
AchieveGoal execution semantics

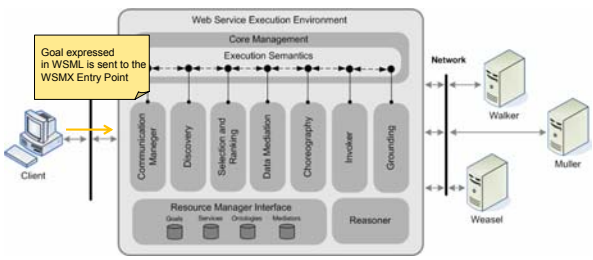


Illustration by larger example
AchieveGoal execution semantics

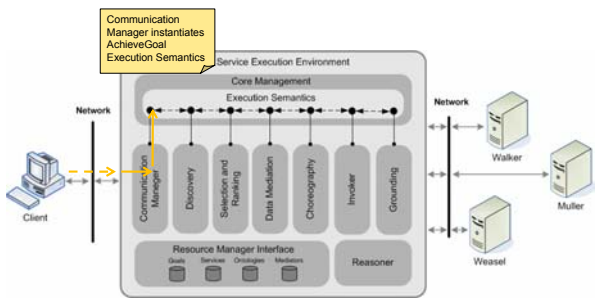


Illustration by larger example
AchieveGoal execution semantics

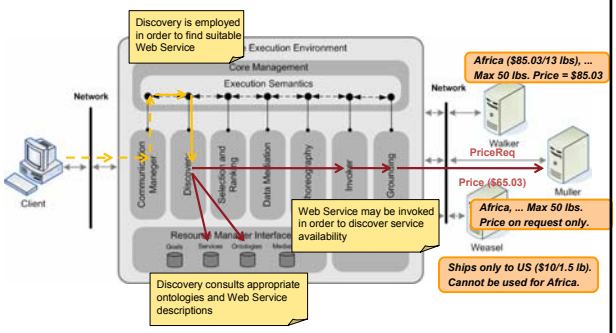


Illustration by larger example
AchieveGoal execution semantics

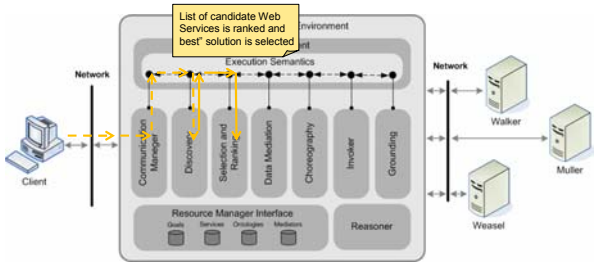


Illustration by larger example
AchieveGoal execution semantics

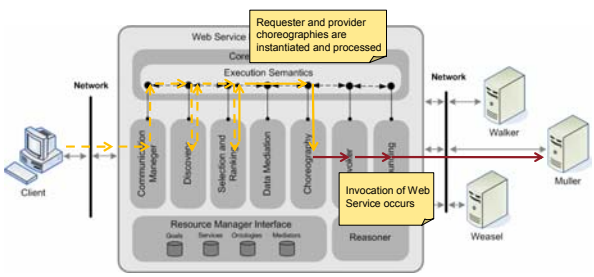


Illustration by larger example
AchieveGoal execution semantics – choreography exec

```
choreography WSMullerShipmentOrderChoreography
   stateSignature WSMullerShipmentOrderStateSignature
   …
   in sop#ShipmentOrderReq withGrounding {_"http://sws-
       challenge.org/shipper/v2/muller.wsdl#wsdl.interfaceMessageReference(muller/ShipmentOrder/in0)"}
   in so#ContactInfo
   in so#ShipmentDate
   in so#Package
   in so#Address
   out sop#ShipmentOrderResp

   transitionRules WSMullerShipmentOrderTransitionRules
   forall {?request} with
      (?request memberOf sop#ShipmentOrderReq)
   do
       add(_#1 memberOf sop#ShipmentOrderResp)
       delete(?request memberOf sop#ShipmentOrderReq)
   endForall
```

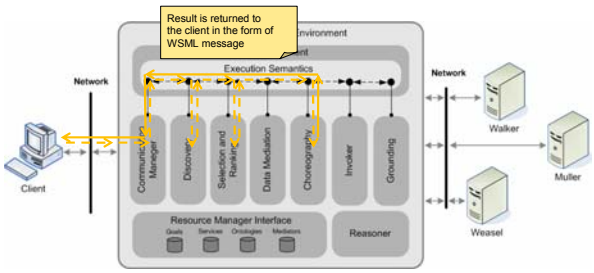
```
<shipmentOrderReq(soi#MoonContactInfo, soi#shipmentDate1, package, soi#SzyslakContactInfo),
  package(1, 7.0, 6.0, 4.0, 1.0),
  shipmentDate1("2009-01-21T13:00:00.046Z", "2009-01-22T13:00:00.046Z")>
```

S1

```
<shipmentOrderResp("2009-01-21T15:00:00.046Z", 65.03),
  package(1, 7.0, 6.0, 4.0, 1.0),
  shipmentDate1("2009-01-21T13:00:00.046Z", "2009-01-22T13:00:00.046Z")>
```

S2

Illustration by larger example
AchieveGoal execution semantics



Summary

- Clear separation of SWS, SESA, SEE and WSMX notions.
- WSMX is reference implementation of WSMO.
- WSMX contributes to solving scalability issues by automating various service related tasks.
- WSMX consists of various components dedicated to solving particular steps in the overall service life cycle orchestrated by Core Management.
- Execution semantics is a formal description of the operational behavior of the system in terms of computational steps

References

- Dieter Fensel, Mick Kerrigan, Michal Zaremba (Eds.), Implementing *Semantic Web Services: The SESA Framework*. Springer Verlag, 2008.
- Charles Petrie, Tiziana Margaria, Holger Lausen, Michal Zaremba (Eds.), *Semantic Web Services Challenge: Results from the First Year*. Springer Verlag, 2008

- Online resources:
  - http://see.sti-innsbruck.at
  - http://sws-challenge.org

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?

STI · INNSBRUCK