Day 3: Semantically Enabled Service Oriented Architectures (SESA)

Omair Shafiq
Digital Enterprise Research Institute (DERI)
Innsbruck, Austria.
Agenda

- Introduction to SESA
- Web Service Execution Environment (WSMX)
  - Technical Insights
- Web Service Modeling Toolkit (WSMT)
- Triple Space Computing (TSC)
  - TSC and WSMX
- WSMX and WSMT Demo
- Hands-on Session
The technology of Semantic Web Services (SWS) envisions:

- Easy access to various systems
- Seamless integration of heterogeneous entities
- Ad-hoc cooperation between various business parties
- Dynamic collaborations on the Web

SEE Vision:

Provide guidelines, justifications and implementation directions for an execution environment for Semantic Web Services
Objectives

Goals:

- Provide a WSMO testbed
- Demonstrate the viability of using WSMO as a means to achieve dynamic inter-operation of Web services

SEE Mission:

Create an execution environment capable of managing all the aspects related to semantically enhanced Web services, to enable their discovery, selection, mediation and execution
Traditional approach to software architecture

“Separate Specialist” model

- No Agility to repair your car even for trivial tasks
- A Process that is duplicative and inefficient
- Costly to operate and maintain – keep many people

In garage every mechanic specialize only in one type of car so it does not matter what you want to repair you always have to wait for a mechanic who knows your type of car; if he/she is sick or on holiday you cannot repair your car at all

Service-Oriented Architecture

“Service-Oriented” model

- Agility to repair cars quickly (next available mechanic takes care)
- A Process that is efficient
- Cost effective to operate and maintain

Mechanic does job himself or asks other mechanics to take care of tasks he is not capable to do

You ask any mechanic in a garage to repair your car – model of your car does not matter
Why SOA?

• In recent years, the decoupling of interface from implementation at the programming level has become a popular architectural approach
  – facilitates the creation of complex systems required by today's business applications
  – interface of a service consumed by a service consumer is loosely coupled with its implementation by a service provider and the implementation is decoupled from its binding.
  – this style has become the key characteristic of Service-Oriented Architecture (SOA)
  – rather than the implementations, only the services provided are published and consumers are insulated from the details of the underlying implementation by a provider or broker.
SOA capabilities

• As a flexible and extensible architectural framework, SOA has the following unique capabilities:

  • **Reduced cost.** Provides the opportunity to consolidate redundant application functionality and decouple functionality from obsolete and increasingly costly applications while leveraging existing investments.
  
  • **Increased flexibility.** Structures IT applications based on services in such a way as to facilitate the rapid restructuring and reconfiguration of the business processes and applications that consume them.
  
  • **Increased revenue.** Provides the opportunity to enter into new markets and leverage existing business capabilities in new and innovative ways using a set of loosely coupled IT services. Helps to increase market share by offering new and better business services.
  
  • **Added agility.** Delivers business-aligned applications faster.
  
  • **Increased consolidation.** Integrates IT systems across siloed applications and organizations.
SOA Reference Architecture
• In the year 2012 software has become a utility, accessed like electrical power through a socket

• How did this revolution happen and who were the driving forces behind it?
Major Objectives

- From software to serviceware
- From syntax to semantics

Tutorial on Semantically Enabled Service Oriented Architectures (SESA)
Objective 1

Moving from Software to Serviceware

- Abstracting from software: All resources are services in a Service Oriented Architecture
- Users are only concerned with the services and not with the software or hardware that implements the service
Objective 2
Moving from Syntax to Semantics

• Add machine-processable semantics to bring Services Oriented Architectures to their full potential

• Only with semantics can critical subtasks be automated, leaving humans to focus on problem solving
Central Hypothesis

Adding semantics to different layers of the SOA will allow us to provide:

- A worldwide networked economy, based on flexible cooperation and outsourcing
- A flexible service innovation and design (see Research Manifesto for Services Science, CACM 49/7, July 2006)
- A reduction of expensive IT maintenance and enhancement of legacy solutions
Scope of our Intervention

- Service-oriented architectures will quickly become the leading software paradigm
- However, SOAs will not scale without significant mechanization of
  - Service discovery, adaptation, negotiation, composition, invocation, and monitoring; and
  - Data and process mediation
- Therefore, machine processable semantics will need to be added to bring SOAs to their full potential
- We are faced with the challenge to move from
  - Software to serviceware and
  - From syntax to semantics
WSMX Introduction

- Software framework for runtime binding of service requesters and service providers
- WSMX interprets service requester’s goal to
  - discover matching services
  - select (if desired) the service that best fits
  - provide mediation (if required)
  - make the service invocation
- Is based on the conceptual model provided by WSMO
- Has a formal execution semantics
- Service Oriented and event-based architecture
  - based on microkernel design using technologies as J2EE, Hibernate, Spring, JMX, etc.
WSMX Motivation

• Middleware ‘glue’ for Semantic Web Services
  – Allow service providers focus on their business
• Reference implementation for WSMO
  – Eat our own cake
• Environment for goal based discovery and invocation
  – Run-time binding of service requester and provider
• 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
• Define formal execution semantics
  – Unambiguous model of system behaviour
WSMX Usage Scenario

Three Tier Architecture

Two Tier Architecture

User (Service Requester)

Client / Browser

WSMX Client

Retailer

Web Server / Application Server

WSMX Server

Manufacturer

WSMX

Back-End Application

Adapter / WSDL Interface

Back-End Application

B2B

B2C
WSMX Usage Scenario - P2P

- A P2P network of WSMX ‘nodes’
- Each WSMX node described as a SWS
- Communication via WSML over SOAP
- Distributed discovery – first aim
- Longer term aim - distributed execution environment
WSMX Usage Scenario - P2P

End User Tools
Developer Tools

Problem Solving Layer
Discovery
Adaptation
Composition
Choreography

Data Mediation
Process Mediation

Communication (external)
Fault Handling
Monitoring

Application Services Layer
Base Services Layer
Storage
Reasoning
Design Principles

• Strong Decoupling & Strong Mediation
  – autonomous components with mediators for interoperability

• Interface vs. Implementation
  – distinguish interface (= description) from implementation (=program)

• Peer to Peer
  – interaction between equal partners (in terms of control)

WSMO Design Principles == WSMX Design Principles == SOA Design Principles
Benefits of SOA

- **Better reuse**
  - Build new functionality (new execution semantics) on top of existing Business Services

- **Well defined interfaces**
  - Manage changes without affecting the Core System

- **Easier Maintainability**
  - Changes/Versions are not all-or-nothing

- **Better Flexibility**
Service Oriented State

- The interface to the service is implementation-independent
- The service can be dynamically invoked
  - Runtime binding
- The service is self-contained
  - Maintains its own state
• Messaging is peer-to-peer facility
• Distributed communication
  – Loosely coupled
• Sender does not need to know receiver (and vice versa)
• Asynchronous mechanism to communicate between software applications
WSMX Architecture

WSMX - Web Services Modeling Toolkit

| WSMX Management | WSMX Monitor | WSML Editor | Ontology Visualizer | Mapping Tools |

WSMX Manager

WSMX Manager Core

CM Wrapper | RM Wrapper | Parser Wrapper | Discovery Wrapper | Selector Wrapper | DM Wrapper | PM Wrapper | Choreography Wrapper

WSMT - Web Services Modeling Toolkit

Service Requesters

Back-End Application

Agent acting on behalf of service requester

Data and Communication Protocols Adapters

Adapter 1

Adapter 2

... Adapter N

Service Providers

Web Service 1

Web Service 2

... Web Service p

WSMO Objects

Non WSMO

Reasoner

Resource Manager Interface

Reasoner Interface

Component Wrapper

Interface

New Component

making semantics real.
Selected Components

- Adapters
- Parser
- Invoker
- Choreography
- Process Mediator
- Discovery
- Data Mediator
- Resource Manager
- Reasoning
1. **Publishing** Create & publish Web service description
2. **Discovery** Determine usable services for a request
3. **Composition** Combine services to achieve a goal
4. **Selection** Choose most appropriate service among the available ones
5. **Mediation** Solve mismatches (data, protocol, process) that hamper interoperation
6. **Execution** Invoke Web services following programmatic conventions
Adapters

- To overcome data representation mismatches on the communication layer
- Transforms the format of a received message into WSML compliant format
- Based on mapping rules
• WSML compliant parser
  – Code handed over to wsmo4j initiative
    http://wsmo4j.sourceforge.net/
• Validates WSML description files
• Compiles WSML description into internal memory model
• Stores WSML description persistently (using Resource Manager)
• WSMX uses
  – The SOAP implementation from Apache AXIS
  – The Apache Web Service Invocation Framework (WSIF)
• WSMO service descriptions are grounded to WSDL
• Both RPC and Document style invocations possible
• Input parameters for the Web Services are translated from WSML to XML using an additional XML Converter component.
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
• 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
Discovery

• Responsible for finding appropriate Web Services to achieve a goal (discovery)

• Discovery component started with providing simple matching
  – Keywords identified in the NFP of the goal
  – Matched against NFPs of the published WSs
  – Variable set of NFPs to be considered for this process

• Using Theorem Prover for Discovery
  – using Theorem Prover such as Vampir, SPASS or Paradox has the advantage of enforcing less restriction during modeling and to support more reasoning tasks compared to Logic Programming Systems

• Discovery Ontology
  – allows more fine grained control over the behavior of the discovery engine
    • specifies that only matches are desired that do not deliver more objects as specified in the goal
    • specifies that if no matches could be found that satisfy the desired logical relationship, that also keyword based matches are accepted
Discovery

- Resource Repository (UDDI or other)
- Keyword- Classification-based Filtering
  - Controlled Vocabulary Filtering
  - Semantic Matchmaking
- usable Web Service
- efficient narrowing of search space (relevant services to be inspected)

retrieve Service Descriptions

invoke Web Service
• Ontology-to-ontology mediation
• A set of mapping rules are defined and then executed
• Initially rules are defined semi-automatic
• Create for each source instance the target instance(s)
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
Design-time Phase

Tutorial on Semantically Enabled Service Oriented Architectures (SESA)

making semantics real.
Design-time Phase - Approach, Decomposition and Mapping Context

- Bottom-up -> training set
- Top-down -> decomposition, context
Run-Time Data Mediator

- 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 a reasoner to evaluate the mapping rules
    - MINS

- Outputs
  - Mediated data
    - Target ontology instances
Ontology Mapping Language

• Language Neutral Mapping Language
  – mapping definitions on meta-layer (i.e. on generic ontological constructs)
  – independent of ontology specification language
  – “Grounding” to specific languages for execution (WSML, OWL, F-Logic)

• Main Features:
  – Mapping Document (sources, mappings, mediation service)
  – direction of mapping (uni- / bidirectional)
  – conditions / logical expressions for data type mismatch handling, restriction of mapping validity, and complex mapping definitions
  – mapping constructs (ex: classMapping, classAttributeMapping, instanceMapping)
  – mapping operators
Mapping Language Example

Ontology O1

Human
- name

Adult

Child

Ontology O2

Person
- name
- age

michael memberOf Person
- name = Michael Stollberg
- age = 28

classMapping(unidirectional o2:Person o1.Adult
attributeValueCondition(o2.Person.age >= 18))

this allows to transform the instance ‘michael’ of concept person in ontology O2 into a valid instance of concept ‘adult’ in ontology O1
• 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
Process Mediator – Addressed mismatches

Diagram: Process Mediation Concepts

- Business Partner 1
- Business Partner 2
- PM (Process Mediator)
- A, B, A and AckA
- Arrows indicate flow of information

Diagram explains the process mediation between business partners, addressing mismatches through a process mediator.
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
    - WS descriptions
    - Mediators
    - Ontologies
  - Event data and intermediate messages
  - WSDL descriptions
Reasoner

- **WSMO4J**
  - validation, serialization and parsing
- **WSML2Reasoner**
  - Reasoning API
    - mapping from WSML to a vendor-neutral rule representation
  - Contains:
    - Common API for WSML Reasoners
    - Transformations of WSML to tool-specific input data (query answering or instance retrieval)
- **WSML-DL-Reasoner**
  - Features:
    - T-Box reasoning (provided by FaCT++)
    - Querying for all concepts
    - Querying for the equivalents, for the children, for the descendants, for the parents and for all ancestors of a given concept
    - Testing the satisfiability of a given concept with respect to the knowledge base
    - Subsumption test of two concepts with respect to the knowledge base
    - Wrapper of WSML-DL to the XML syntax of DL used in the DIG interface

- **Mins**
  - Datalog + Negation + Function Symbols Reasoner Engine
  - Features
    - Built-in predicates
    - Function symbols
    - Stratified negation

- Other Reasoner Engines, e.g. Kaon2, DLV and Flora

- WSMO4J
  - Reasoning API
  - Contains:
    - Built-in predicates
    - Function symbols
    - Stratified negation

- WSML2Reasoner
  - Reasoning API
  - Features:
    - T-Box reasoning (provided by FaCT++)
    - Querying for all concepts
    - Querying for the equivalents, for the children, for the descendants, for the parents and for all ancestors of a given concept
    - Testing the satisfiability of a given concept with respect to the knowledge base
    - Subsumption test of two concepts with respect to the knowledge base
    - Wrapper of WSML-DL to the XML syntax of DL used in the DIG interface
• `achieveGoal` (WSMLDocument): Context

• `getWebServices` (WSMLDocument): Context

• `invokeWebService` (WSMLDocument, Context): Context
Define “Business” Process

- Discover Web Services
- Start
- Create Choreography
- Created
- Discover Services
- Mediate Data
- Mediate Data
- Return Mediated Data
- Return Mediated Data
- Return Web Services
- Check Choreography
- Confirmed
- Call invoker
- Confirmed
- End
Generate Wrappers for Components

- Registry of known components
- Discovery Wrapper
- Choreography Wrapper
- Communication Manager Wrapper
- Start
  - Create Choreography
  - Created
  - Discover Web Services
    - Mediate Data
      - Mediate Data
      - Return Mediated Data
    - Return Web Services
      - Check Choreography
        - Confirmed
        - Confirmed
        - End
  - Discover Services
    - Mediate Data
Context Data

REGISTRY OF KNOWN COMPONENTS

DISCOVERY WRAPPER

DATA MEDIATOR

COMMUNICATION MANAGER WRAPPER

CHOREOGRAPHY OBJECT
Mediated objects, Web Services entities

ERRORS
Exceptions

PROCESS CONTEXT

DISCOVER WEB SERVICES
Start

CREATE CHOREOGRAPHY
Created

DISCOVER SERVICES

MEDIATE DATA

RETURN MEDIATED DATA

RETURN WEB SERVICES

CHECK CHOREOGRAPHY
Confirmed

CALL INVOCER
Confirmed

END

CHOREOGRAPHY WRAPPER
Event-based Implementation

Core - Manager

“Business” Process – Internal Workflow

Event and Notification Distribution/Delivery Mechanism

Choreography Wrapper

Discovery Wrapper Implements Mediator Interface

Usda Mediator Wrapper

Communication Manager Wrapper
Request to discover Web services.
Execution Semantics

Goal expressed in WSML is sent to WSMX System Interface

WSMX Manager

WSMX Manager Core

CM Wraper
RM Wraper
Parser Wraper
Discovery Wraper
Selector Wraper
DM Wraper
PM Wraper
Choreography Wraper

Interface
Interface
Interface
Interface
Interface
Interface
Interface
Interface

Communication Manager
Resource Manager
Parser
Discovery
Selector
Data Mediator
Process Mediator
Choreography

Invoker
Receiver
Grounding

Resource Manager Interface

WSMO Objects
Non WSMO

Reasoner Interface

Reasoner

Component Wrapper
Interface
New Component

WSMT – Web Services Modelling Toolkit

WSMX Monitor
WSML Editor
Choreography Editor
Mediator Editor

Administration Framework Interface

Service Requesters
Service Providers

Back-End Application
Agent acting on behalf of service requester

Web Service 1
Web Service 2
...
Web Service p

WSML Editor
WSMX Monitor
Choreography Editor
Mediator Editor

WSMO Objects
Reasoner

New Component

Service Requesters
Service Providers

Goal expressed in WSML is sent to WSMX System Interface

WSMX Manager

WSMX Manager Core

CM Wraper
RM Wraper
Parser Wraper
Discovery Wraper
Selector Wraper
DM Wraper
PM Wraper
Choreography Wraper

Interface
Interface
Interface
Interface
Interface
Interface
Interface
Interface

Communication Manager
Resource Manager
Parser
Discovery
Selector
Data Mediator
Process Mediator
Choreography

Invoker
Receiver
Grounding

Resource Manager Interface

WSMO Objects
Non WSMO

Reasoner Interface

Reasoner

Component Wrapper
Interface
New Component

WSMT – Web Services Modelling Toolkit

WSM X Monitor
WSML Editor
Choreography Editor
Mediator Editor

Administration Framework Interface

Service Requesters
Service Providers

Back-End Application
Agent acting on behalf of service requester

Web Service 1
Web Service 2
...
Web Service p

WSML Editor
WSMX Monitor
Choreography Editor
Mediator Editor

WSMO Objects
Reasoner

New Component
Com. M. implements the interface to receive WSML goals.
WSMT – Web Services Modelling Toolkit

WSMX Manage | WSMX Monitor | WSML Editor | Choreography Editor | Mediator Editor

WSMX Manager Core

Com. M. informs Core that Goal has been received

WSMX Manager

Resource Manager Interface

WSMO Objects | Non WSMO

Reasoner Interface

Reasoner

New Component

Component Wrapper

Interface

Invoker Receiver

Grounding

WSMX Interface

CM Wrapper

Resource Manager Interface

PM Wrapper

Process Mediator

Choreography Mediator

Selector Wrapper

DM Wrapper

Data Mediator

Discovery Wrapper

RM Wrapper

Selector

Interface

WSMX Monitor

Administration Framework Interface

WSMO Objects

Non WSMO

Reasoner

New Component

Component Wrapper

Interface

Invoker Receiver

Grounding

WSMX Interface

CM Wrapper

Resource Manager Interface

PM Wrapper

Process Mediator

Choreography Mediator

Selector Wrapper

DM Wrapper

Data Mediator

Discovery Wrapper

RM Wrapper

Selector

Interface

WSM - Web Services Modelling Toolkit

WSMX Manage | WSMX Monitor | WSML Editor | Choreography Editor | Mediator Editor

WSMX Manager Core

Com. M. informs Core that Goal has been received

WSMX Manager

Resource Manager Interface

WSMO Objects | Non WSMO

Reasoner Interface

Reasoner

New Component

Component Wrapper

Interface

Invoker Receiver

Grounding

WSMX Interface

CM Wrapper

Resource Manager Interface

PM Wrapper

Process Mediator

Choreography Mediator

Selector Wrapper

DM Wrapper

Data Mediator

Discovery Wrapper

RM Wrapper

Selector

Interface

WSM - Web Services Modelling Toolkit

WSMX Manage | WSMX Monitor | WSML Editor | Choreography Editor | Mediator Editor

WSMX Manager Core

Com. M. informs Core that Goal has been received

WSMX Manager

Resource Manager Interface

WSMO Objects | Non WSMO

Reasoner Interface

Reasoner

New Component

Component Wrapper

Interface

Invoker Receiver

Grounding

WSMX Interface

CM Wrapper

Resource Manager Interface

PM Wrapper

Process Mediator

Choreography Mediator

Selector Wrapper

DM Wrapper

Data Mediator

Discovery Wrapper

RM Wrapper

Selector

Interface

WSM - Web Services Modelling Toolkit

WSMX Manage | WSMX Monitor | WSML Editor | Choreography Editor | Mediator Editor

WSMX Manager Core

Com. M. informs Core that Goal has been received

WSMX Manager

Resource Manager Interface

WSMO Objects | Non WSMO

Reasoner Interface

Reasoner

New Component

Component Wrapper

Interface

Invoker Receiver

Grounding

WSMX Interface

CM Wrapper

Resource Manager Interface

PM Wrapper

Process Mediator

Choreography Mediator

Selector Wrapper

DM Wrapper

Data Mediator

Discovery Wrapper

RM Wrapper

Selector

Interface

WSM - Web Services Modelling Toolkit

WSMX Manage | WSMX Monitor | WSML Editor | Choreography Editor | Mediator Editor

WSMX Manager Core

Com. M. informs Core that Goal has been received

WSMX Manager

Resource Manager Interface

WSMO Objects | Non WSMO

Reasoner Interface

Reasoner

New Component

Component Wrapper

Interface

Invoker Receiver

Grounding

WSMX Interface

CM Wrapper

Resource Manager Interface

PM Wrapper

Process Mediator

Choreography Mediator

Selector Wrapper

DM Wrapper

Data Mediator

Discovery Wrapper

RM Wrapper

Selector

Interface

WSM - Web Services Modelling Toolkit

WSMX Manage | WSMX Monitor | WSML Editor | Choreography Editor | Mediator Editor

WSMX Manager Core

Com. M. informs Core that Goal has been received

WSMX Manager

Resource Manager Interface

WSMO Objects | Non WSMO

Reasoner Interface

Reasoner

New Component

Component Wrapper

Interface

Invoker Receiver

Grounding

WSMX Interface

CM Wrapper

Resource Manager Interface

PM Wrapper

Process Mediator

Choreography Mediator

Selector Wrapper

DM Wrapper

Data Mediator

Discovery Wrapper

RM Wrapper

Selector

Interface

WSM - Web Services Modelling Toolkit

WSMX Manage | WSMX Monitor | WSML Editor | Choreography Editor | Mediator Editor

WSMX Manager Core

Com. M. informs Core that Goal has been received

WSMX Manager

Resource Manager Interface

WSMO Objects | Non WSMO

Reasoner Interface

Reasoner

New Component

Component Wrapper

Interface
WSMX – Web Services Modelling Toolkit

WSMX Manager

WSMX Manager Core

CM Wrapper
RM Wrapper
Parser Wrapper
Discovery Wrapper
Selector Wrapper
DM Wrapper
PM Wrapper
Choreography Wrapper

Communication Manager
Resource Manager
Parser
Discovery
Selector
Data Mediator
Process Mediator
Choreography

Resource Manager Interface
WSMO Objects
Non WSMO

Reasoner Interface
Reasoner

WSM - System Interface
Client
Server
Adapter 1
Adapter 2
... Adapter n

Service Requesters
Back-End Application
Agent acting on behalf of service requester

Service Providers
Web Service 1
Web Service 2
Web Service p

Execution Semantics

Chor. wrapper picks up event for Chor. component

Component Wrapper
Interface
New Component
WSMX System Interface
WSMX Manager
WSMX Manager Core
Administration Framework Interface
Grounding
CM Wrapper
Communication Manager
Invoker
Receiver
Grounding
Interface
RM Wrapper
Resource Manager
Parser
Interface
Discovery Wrapper
Discovery
Interface
Selector
Selector
Interface
Data
Data Mediator
Interface
Process Mediator
Interface
Choreography
Choreography
Interface
Component Wrapper
Component Wrapper
Interface
New Component
WSMT – Web Services Modelling Toolkit
WSMX Management
WSMX Monitor
WSML Editor
Choreography Editor
Mediator Editor
WSMO Objects
Reasoner
Reasoner Interface
Non WSMO
New choreography instance is created
Service Requesters
Service Providers
Back-End Application
Agent acting on behalf of service requester
Adapter 1
Adapter 2
... Adapters
... Adapters
WSDL Editor
WSMX Monitor
Choreography Editor
Mediator Editor
Web Service 1
Web Service 2
... Web Service p
Web Service 1
Web Service 2
... Web Service p
Service Requesters
Service Providers
Back-End Application
Agent acting on behalf of service requester
Adapter 1
Adapter 2
... Adapters
WSDL Editor
WSMX Monitor
Choreography Editor
Mediator Editor
Web Service 1
Web Service 2
... Web Service p
Web Service 1
Web Service 2
... Web Service p
Service Requesters
Service Providers
Back-End Application
Agent acting on behalf of service requester
Adapter 1
Adapter 2
... Adapters
WSDL Editor
WSMX Monitor
Choreography Editor
Mediator Editor
Web Service 1
Web Service 2
... Web Service p
Web Service 1
Web Service 2
... Web Service p
Service Requesters
Service Providers
Back-End Application
Agent acting on behalf of service requester
Adapter 1
Adapter 2
... Adapters
WSDL Editor
WSMX Monitor
Choreography Editor
Mediator Editor
Web Service 1
Web Service 2
... Web Service p
Web Service 1
Web Service 2
... Web Service p
Execution Semantics

WSMX – Web Services Modelling Toolkit

WSMX Manager

WSMX Manager Core

CM Wrapper
RM Wrapper
Parser Wrapper
Discovery Wrapper
Selector Wrapper
DM Wrapper
PM Wrapper
Choreography Wrapper

CM Wrapper Interface
RM Wrapper Interface
Interface
Parser Interface
Discovery Interface
Selector Interface
DM Interface
PM Interface
Choreography Interface

Resource Manager Interface

WSMO Objects
Non WSMO
Reasoner

Reasoner Interface

Component Wrapper
Interface
New Component

Execution Semantics
WSMX – Web Services Modelling Toolkit

WSMX Management
WSMX Monitor
WSML Editor
Choreography Editor
Mediator Editor

Service Requesters
Back-End Application
Agent acting on behalf of service requester

Service Providers
Web Service 1
Web Service 2
Web Service p

WSMX Manager
WSMX Manager Core

Reasoner Interface
Choreography Interface
Process Mediator Interface
Data Mediator Interface
Selector Interface
Discovery Interface
Parser Interface
Invoker Interface
CM Wrapper
RM Wrapper
Selector Wrapper
Discovery Wrapper
Parser Wrapper

WSML goal is parsed to internal format.

Resource Manager Interface
WSMO Objects
Non WSMO

Reasoner Interface
Reasoner
Component Wrapper
New Component

WSML – Web Services Modelling Toolkit
Discovery is invoked for parsed goal.
WSMX: System Interface
WSMX Manager
WSMX Manager Core
CM Wrapper
RM Wrapper
Parser Wrapper
Discovery Wrapper
Selector Wrapper
DM Wrapper
PM Wrapper
Choreography Wrapper
Interface
Communication Manager
Resource Manager
Parser
Discovery
Selector
Discovery
Process Mediator
Choreography
Component Wrapper
New Component
WSMT – Web Services Modelling Toolkit
WSMX Management
WSMX Monitor
WSML Editor
Choreography Editor
Mediator Editor
Administration Framework Interface
WSMO Objects
Non WSMO
Reasoner
Discovery may requires ontology mediation.
Execution Semantics

After data mediation, Discovery iterates, if needed through last steps until result set is finished.

WSMX Manager Core
- CM Wrapper
- RM Wrapper
- Parser Wrapper
- Discovery Wrapper
- Selector Wrapper
- DM Wrapper
- PM Wrapper
- Choreography Wrapper

WSMX Manager
- Administration Framework Interface
- CM Wrapper
- RM Wrapper
- Parser Wrapper
- Discovery Wrapper
- Selector Wrapper
- DM Wrapper
- PM Wrapper
- Choreography Wrapper

WSMT – Web Services Modelling Toolkit
- WSMX Management
- WSMX Monitor
- WSML Editor
- Choreography Editor
- Mediator Editor

Service Requesters
- Back-End Application
- Agent acting on behalf of service requester
- Data and Communication Protocols Adapters
  - Adapter 1
  - Adapter 2
  - Adapter n

Service Providers
- Web Service 1
- Web Service 2
- Web Service p

Component Wrapper
- New Component

WSMO Objects
Non WSMO
Reasoner

After data mediation, Discovery iterates, if needed through last steps until result set is finished.
Execution Semantics

WSM – Web Services Modelling Toolkit

WSMX Manager

WSMX Manager Core

CM Wrapper
RM Wrapper
Parsar Wrapper
Discovery Wrapper
Selector Wrapper
DM Wrapper
PM Wrapper
Choreography Wrapper

Interface
Interface
Interface
Interface
Interface
Interface
Interface
Interface

CM Wrapper
RM Wrapper
Parsar Wrapper
Discovery Wrapper
Selector Wrapper
DM Wrapper
PM Wrapper
Choreography Wrapper

Invoker
Receiver
Grounding

Resource Manager Interface

WSMO Objects
Non WSMO

Reasoner Interface

Reasoner

Component Wrapper

New Component

Selection is invoked to relax result set to finally one service.

Service Requesters
Back-End Application
Agent acting on behalf of service requester

Data and Communication Protocols Adapters
Adapter 1
Adapter 2
... Adapter n

Service Providers
Web Service 1
Web Service 2
... Web Service p

WSML Editor
WSMX Monitor
WSMO Objects
WSMT – Web Services Modelling Toolkit

Service Providers
Web Service 1
Web Service 2
... Web Service p

WSML Editor
WSMX Monitor
WSMO Objects
WSMT – Web Services Modelling Toolkit

Service Requesters
Back-End Application
Agent acting on behalf of service requester

Data and Communication Protocols Adapters
Adapter 1
Adapter 2
... Adapter n

Execution Semantics

WSM – Web Services Modelling Toolkit

WSMX Manager

WSMX Manager Core

CM Wrapper
RM Wrapper
Parsar Wrapper
Discovery Wrapper
Selector Wrapper
DM Wrapper
PM Wrapper
Choreography Wrapper

Interface
Interface
Interface
Interface
Interface
Interface
Interface
Interface

CM Wrapper
RM Wrapper
Parsar Wrapper
Discovery Wrapper
Selector Wrapper
DM Wrapper
PM Wrapper
Choreography Wrapper

Invoker
Receiver
Grounding

Resource Manager Interface

WSMO Objects
Non WSMO

Reasoner Interface

Reasoner

Component Wrapper

New Component

Selection is invoked to relax result set to finally one service.
WSMT – Web Services Modelling Toolkit

WSMX Management
WSMX Monitor
WSML Editor
Choreography Editor
Mediator Editor

WSMX
Administration Framework Interface

WSMX Manager
WSMX Manager Core

CM Wrapper
RM Wrapper
Parser Wrapper
Discovery Wrapper
Selector Wrapper
DM Wrapper
PM Wrapper
Choreography Wrapper

Interface
Interface
Interface
Interface
Interface
Interface
Interface
Interface

Communication Manager
Resource Manager
Parser
Discovery
Selector
Data Mediator
Process Mediator

Invoker
Receiver
Grounding

Choreography instance for goal requester is checked for next steps.

WSMO Objects
Non WSMO
Reasoner

Resource Manager Interface
Reasoner Interface

WSML Editor
Choreography Editor
WSMX Management
Mediator Editor

Service Requesters
Back-End Application
Agent acting on behalf of service requester

Adapter 1
Adapter 2
Adapter n

Service Providers
Web Service 1
Web Service 2
Web Service p

WSML Editor
WSMX Monitor
WSMX Management
Choreography Editor
Mediator Editor

WSMO Objects
Non WSMO
Reasoner

Resource Manager Interface
Reasoner Interface

Execution Semantics
Execution Semantics

WSMX Manager

WSMX Manager Core

Resource Manager Interface

Reasoner Interface

WSMO Objects

Non WSMO

Reasoner

WSMT – Web Services Modelling Toolkit

WSMX Management
WSMX Monitor
WSML Editor
Choreography Editor
Mediator Editor

WSMX

Grounding

CM Wrapper
RM Wrapper
Parasr Wrapper
Discovery Wrapper
Selector Wrapper
DM Wrapper
PM Wrapper
Choreography Wrapper

Interface

Parser
Discovery
Selector
Data Mediator
Process Mediator
Choreography

Component Wrapper

New Component

Result is returned to Com. Man. to be forwarded to the service requester.

Execution Semantics

Client

Server

Service Requesters

Data and Communication Protocols Adapters

Service Providers

WSML Editor
WSMX Monitor
WSMX Management
Choreography Editor
Mediator Editor

Web Service 1
Web Service 2
Web Service p

Component Wrapper

New Component

WSMT – Web Services Modelling Toolkit

WSMX Management
WSMX Monitor
WSML Editor
Choreography Editor
Mediator Editor

WSMX

Grounding

CM Wrapper
RM Wrapper
Parasr Wrapper
Discovery Wrapper
Selector Wrapper
DM Wrapper
PM Wrapper
Choreography Wrapper

Interface

Parser
Discovery
Selector
Data Mediator
Process Mediator
Choreography

Result is returned to Com. Man. to be forwarded to the service requester.
Execution Semantics

Set of Web Service descriptions expressed in WSML sent to adapter.

WSMX – Web Services Modelling Toolkit
- WSMT – Web Services Modelling Toolkit
- WSMX Manager
- WSMX Monitor
- WSML Editor
- Choreography Editor
- Mediator Editor

WSMX Manager Core
- CM Wrapper
- RM Wrapper
- Parser Wrapper
- Discovery Wrapper
- Selector Wrapper
- DM Wrapper
- PM Wrapper
- Choreography Wrapper

Service Requesters
- Back-End Application
- Agent acting on behalf of service requester

Data and Communication Protocols Adapters
- Adapter 1
- Adapter 2
- Adapter n

Service Providers
- Web Service 1
- Web Service 2
- Web Service p

Resource Manager Interface
- WSMO Objects
- Non WSMO

Reasoner Interface
- Reasoner

Component Wrapper
- Interface
- New Component

CM Wrapper
RM Wrapper
Parser Wrapper
Discovery Wrapper
Selector Wrapper
DM Wrapper
PM Wrapper
Choreography Wrapper

Invokers
Invokers
Invokers
Invokers
Invokers
Invokers
Invokers
Invokers

WSML Editor
WSMX Monitor
Choreography Editor
Mediator Editor

Grounding

Set of Web Service descriptions expressed in WSML sent to adapter.

Execution Semantics
Execution Semantics

Set of Web Service descriptions expressed in requester's own format returned to goal requester.

WSMX Manager

WSMX Manager Core

- CM Wrapper
- RM Wrapper
- Parser Wrapper
- Discovery Wrapper
- Selector Wrapper
- DM Wrapper
- PM Wrapper
- Choreography Wrapper

WSMT – Web Services Modelling Toolkit

- WSMX Management
- WSMX Monitor
- WSML Editor
- Choreography Editor
- Mediator Editor

Administration Framework Interface

WSML Editor

WSMX Monitor

Choreography Editor

Mediator Editor

Requesters

- Back-End Application
- Agent acting on behalf of service requester

Data and Communication Protocols Adapters

- Adapter 1
- Adapter 2
- Adapter n

Service Providers

- Web Service 1
- Web Service 2
- Web Service p

Requesters

- Invoker
- Receiver

Grounding

Resource Manager Interface

- WSMO Objects
- Non WSMO

Reasoner Interface

- Reasoner

Component Wrapper

New Component

Set of Web Service descriptions expressed in requester's own format returned to goal requester.

WSMO Objects

Non WSMO

Reasoner

Component Wrapper

New Component

Execution Semantics

Set of Web Service descriptions expressed in requester's own format returned to goal requester.

WSMX Manager

WSMX Manager Core

- CM Wrapper
- RM Wrapper
- Parser Wrapper
- Discovery Wrapper
- Selector Wrapper
- DM Wrapper
- PM Wrapper
- Choreography Wrapper

WSMT – Web Services Modelling Toolkit

- WSMX Management
- WSMX Monitor
- WSML Editor
- Choreography Editor
- Mediator Editor

Administration Framework Interface

WSML Editor

WSMX Monitor

Choreography Editor

Mediator Editor

Requesters

- Back-End Application
- Agent acting on behalf of service requester

Data and Communication Protocols Adapters

- Adapter 1
- Adapter 2
- Adapter n

Service Providers

- Web Service 1
- Web Service 2
- Web Service p

Requesters

- Invoker
- Receiver

Grounding

Resource Manager Interface

- WSMO Objects
- Non WSMO

Reasoner Interface

- Reasoner

Component Wrapper

New Component
Web Service Modeling Toolkit

WSMT

SESA Tutorial,
The aim of the Web Services Modeling Toolkit (WSMT) is to provide high-quality tools for designing, mediating and using Semantic Web Services, through the WSMO paradigm.

The focus is currently on the following areas:

- Creation of ontologies, web services, goals and mediators in WSMO
- Creation of mappings between pairs of ontologies to allow runtime instance transformation
- Management of Execution Environments for Semantic Web Services like WSMX and IRSIII
• Perspectives in the Eclipse framework allow for a number of Editors and views to be grouped and positions.

• The WSML perspective offers editors and views related to engineering of semantic descriptions in WSMO through the WSML language.

• Other General features include:
  – WSML file validation
  – Problems view (errors and warnings on files in the workspace)
  – Label highlighting (marking of errors and warnings in navigator view)
Editors
WSML Text Editor
WSML Conceptual Editor
WSML Visualizer

Views
Navigator view
Problems view
WSML Reasoner
WSML Perspective: Editors & Views

Editors
- WSML Text Editor
- WSML Conceptual Editor
- WSML Visualizer

Views
- Navigator view
- Problems view
- WSML Reasoner
Editors
WSML Text Editor
WSML Conceptual Editor
WSML Visualizer

Views
Navigator view
Problems view
WSML Reasoner
WSML Perspective: Editors & Views

Editors
WSML Text Editor
WSML Conceptual Editor
WSML Visualizer

Views
Navigator view
Problems view
WSML Reasoner
WSML Perspective: Editors & Views

Editors
WSML Text Editor
WSML Conceptual Editor
WSML Visualizer

Views
Navigator view
Problems view
WSML Reasoner
WSML Perspective: Editors & Views

Views
Navigator view
Problems view
WSML Reasoner

Table:

<table>
<thead>
<tr>
<th>Description</th>
<th>Resource</th>
<th>In Folder</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Could not parse WSML: (Line: 508 Pos: 1 Expected: '&quot;', '}', 'false', 'true', 'false', '...</td>
<td>amazon-ontology.wsml</td>
<td>amazon</td>
<td>line 508</td>
</tr>
<tr>
<td>Could not parse WSML: (Line: 15 Pos: 1 Expected: EOF Found:onto)</td>
<td>amazon-webservice.wsml</td>
<td>amazon</td>
<td>line 15</td>
</tr>
</tbody>
</table>
WSML Perspective: Editors & Views

Editors
WSML Text Editor
WSML Conceptual Editor
WSML Visualizer

Views
Navigator view
Problems view
WSML Reasoner
Abstract Mapping Language: Editors & Views

Editors
- AML Text Editor
- AML Conceptual Editor
- AML View Based Editor

Views
- Concept 2 Concept View
- Attribute 2 Attribute View
- Concept 2 Attribute View
- Attribute 2 Concept View
- Status View
Editors
AML Text Editor
AML Conceptual Editor
AML View Based Editor

Views
Concept 2 Concept View
Attribute 2 Attribute View
Concept 2 Attribute View
Attribute 2 Concept View
Status View
Editors
AML Text Editor
AML Conceptual Editor
AML View Based Editor

Views
Concept 2 Concept View
Attribute 2 Attribute View
Concept 2 Attribute View
Attribute 2 Concept View
Status View
Views

Concept 2 Concept View
Attribute 2 Attribute View
Concept 2 Attribute View
Attribute 2 Concept View
Status View
### Abstract Mapping Language: Editors & Views

#### Views

- **Concept 2 Concept View**
- **Attribute 2 Attribute View**
- **Concept 2 Attribute View**
- **Attribute 2 Concept View**
- **Status View**

<table>
<thead>
<tr>
<th>Source Attributes</th>
<th>Target Attributes</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>[(date) day =&gt; integer]</td>
<td>[(date) day =&gt; integer]</td>
<td>No conditions associated</td>
</tr>
<tr>
<td>[(date) month =&gt; integer]</td>
<td>[(date) month =&gt; integer]</td>
<td>No conditions associated</td>
</tr>
<tr>
<td>[(date) year =&gt; integer]</td>
<td>[(date) year =&gt; integer]</td>
<td>No conditions associated</td>
</tr>
</tbody>
</table>
### Views

**Concept 2 Concept View**

**Attribute 2 Attribute View**

**Concept 2 Attribute View**

**Attribute 2 Concept View**

**Status View**

---

**Abstract Mapping Language: Editors & Views**

<table>
<thead>
<tr>
<th>Source Concepts</th>
<th>Target Attributes</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ticket</td>
<td>(travelVoucher) bearer =&gt; name</td>
<td>No conditions associated</td>
</tr>
</tbody>
</table>
Views

Concept 2 Concept View
Attribute 2 Attribute View
Concept 2 Attribute View
Attribute 2 Concept View
Status View
Views
Concept 2 Concept View
Attribute 2 Attribute View
Concept 2 Attribute View
Attribute 2 Concept View
Status View
Conclusions

- Conceptual model is WSMO
- End to end functionality for executing SWS
- Has a formal execution semantics
- Real implementation
- Open source code base at SourceForge
- Event-driven component architecture
- WSMT – emerging tool to handle semantics
Triple Space Computing results from the integration of three existing and well known technologies.

“Triple Spaces shall embody a communication paradigm for anonymous, asynchronous, scalable information exchange through publication that ensures persistency and unique identification (URI) of the communicated semantic data”

“Triple Space shall become the web for machines as the web based on HTML became the Web for humans"
Decoupling of Interaction

- SBC decouples interaction in three dimensions
  - **Time**
    - Applications can read and write data whenever they want to
  - **Space**
    - Applications only need to access the same space in order to communicate
  - **Reference**
    - Applications communicating with each other do not need to know explicitly from each other
write, read, take, query, update, subscribe, advertise based on
- URIs
- Templates
  (cf. SPARQL, N3QL)
- Named Graphs

setPermissions, createUser, deleteUser, createRole, deleteRole, addUserToRole based on
- Named Graphs
- Meta Graph

add, remove mediation rules based on
- RDF Triples
- URIs
Mediation-check based on
- Templates
• Triple Space Computing for Semantic Web Services
  – Dynamic Components Management in WSMX using Triple Space
  – Inter-WSMX communication and coordination using Triple Space
  – Resource Management in WSMX using Triple Space
  – External Communication with WSMX using Triple Space
Message based coordination of components in WSMX

Web Service Execution Environment (WSMX)

WSMX Manager
Discovery
Selection
Composition
Data Mediation
Process Mediation
Communication Manager
Components Management of WSMX using TSC

Web Service Execution Environment (WSMX)

- Discovery Interface
- Selection Interface
- Composition Interface
- WSMX Manager Interface
- Data Mediation Interface
- Parser Interface
- Other Components Interface

Wrapper

TS Proxy

Composition

Wrapper

TS Proxy

Selection

Wrapper

TS Proxy

Interface

Data Mediation

Wrapper

TS Proxy

Parser

Wrapper

TS Proxy

Other Components

Wrapper

TS Proxy

Discoveries, Selections, and Compositions

- Ts Proxy

Wrappers

Parser

Web Service Execution Environment (WSMX)

TSC API

Security Management API

Mediation Management API

Operation Layer

Security

Coordination Layer

Data Access API

Data Access API

YARS
Components Management of WSMX using TSC

Web Service Execution Environment (WSMX)

Discovery Interface
Selection Interface
Composition Interface
WSMX Manager Interface
Data Mediation Interface
Parser Interface
Other Components Interface

Wrapper
TS Proxy
Wrapper
TS Proxy
Wrapper
TS Proxy
Wrapper
TS Proxy
Wrapper
TS Proxy
Wrapper
TS Proxy
Wrapper
TS Proxy

Triple Space Computing Middleware
TSC for inter-connecting multi WSMXs

- Inter-connecting different WSMX instances running at different places
- Triple Space as global communication middleware
Interfacing WSMX resource manager with Triple Space Kernel
Passing the data received by Resource Manager Layer to TS Kernel
Bringing TSC based Resource Management for WSMX
  - Reduced complexity for WSMX resource manager to contains internal repositories
  - Help in Distribution of Resources for other WSMXs
  - Improving time to access the required resources
TSC based External Communication in WSMX

- Service Requestor submitting Goal to WSMX
  - Web Service end-points
- Providing additional communication channel
  - Submitting Goal via Triple Space
  - Decoupling external applications and WSMX
- Grounding Goals to Triple Space
- Interfacing Communication Manager with TS Kernel
Hands-on Session
WSMT Demo

SESA Tutorial,
WSMX Demo

SESA Tutorial,

© Copyright 2007  DERI Innsbruck  www.deri.at

making semantics real.
Use Case

• **Virtual Travel Agency**
  – Provides flight and train services

• **Service Provider**
  – Austrian Air
  – US Air
  – Oebb (Austrian Railways)

• **WSMX**
  – Acting as middleware to bring together **Clients** and the **Virtual Travel Agency** to find the best combination of **services** for clients
Use case Scenario

Virtual Travel Agency (VTA)

- Passenger
- Austrian Air
- US Air
- OeBB Austrian Train
Why WSMX?

• Discovery
  – Client/Passenger does not know which airline goes to his/her destination
  – Which is the suitable option (i.e. train or flight)
  – Client/Passenger does not want to bound to one provider

• Mediation
  – Client uses a different format of data than that of Airlines and Train reservation Web Services
  – Client does not want to change the data format

• WSMX acts as middleware between the client and end-point Web Services
Discovery Scenario

• Searching All possible services
  – Providing train based travel
  – Providing flights based travel
  – Traveling to US
  – Traveling to EU
  – Traveling from Innsbruck (Austria) to Vienna (Austria)
• Heterogeneity between the user and service provider data

<table>
<thead>
<tr>
<th>User</th>
<th>Service Provider</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>Payment</td>
</tr>
<tr>
<td>Ticket</td>
<td>Travel Voucher</td>
</tr>
<tr>
<td>Currency</td>
<td>Payment</td>
</tr>
<tr>
<td>Train direction and time table</td>
<td>Trip points</td>
</tr>
<tr>
<td>Terms (payment method)</td>
<td>Payment type</td>
</tr>
</tbody>
</table>
Thank you for your attention