Adaptable Service Oriented Architectures
SOA
Service Oriented Architecture
Attributes of physical services

• Well defined, easy-to-use, somewhat standardized interface
• Self-contained with no visible dependencies to other services
• (almost) Always available but idle until requests come
• “Provision-able”
• Easily accessible and usable readily, no “integration” required
• Coarse grain
• Independent of consumer context,
  – but a service can have a context
• New services can be offered by combining existing services
• Quantifiable quality of service
  – Do not compete on “What” but “How”
  – Performance/Quality
  – Cost
  – ...
Context, Composition and State

• Services are most often designed to ignore the context in which they are used
  – It does not mean that services are stateless they are rather context independent!
  – This is precisely the definition of “loosely coupled”
    • Services can be reused in contexts not known at design time
• Value can be created by combining, i.e. “composing” services
  – Book a trip versus book a flight, car, hotel, …
Service Interfaces

- Non proprietary
  - All service providers offer somewhat the same interface
- Highly Polymorphic
  - Intent is enough
- Implementation can be changed in ways that do not break all the service consumers
  - Real world services interact with thousands of consumers
  - Service providers cannot afford to “break” the context of their consumers
Intents and Offers

• Service consumer expresses “intent”
• Service providers define “offers”

• Sometimes a mediator will:
  – Find the best offer matching an intent
  – Advertise an offer in different ways such that it matches different intent

• Request / Response is just a very particular case of an Intent / Offer protocol
Service Orientation and Directories

• Our society could not be “service oriented” without the “Yellow Pages”
• Directories and addressing mechanisms are at the center of our service oriented society
• Imagine
  – Being able to reach a service just by using longitude and latitude coordinates as an addressing mechanism?
  – Only being able to use a service if you can remember its location, phone or fax number?
Service Orientation is scalable

• Consumers can consume and combine a lot of services since they don’t have to know or “learn” how to use a service

• Service providers can offer their services to a lot more consumers because by optimizing
  – The user interface
  – Access (Geographical, Financial, …)
  – They were able to provide the best quality of service and optimize their implementations
So…

• Service orientation allows us
  – Complete freedom to create contexts in which services are uses and combined
  – Express intent rather than specific requests

• Our society should be a great source of inspiration to design modern enterprise systems and architectures or understand what kind of services these systems will consume or provide
Connectivity Enables Global Processes and Information Access

- **Connectivity Evolution**:
  - **1980**: LAN
  - **1990**: Web
  - **2000**: XML
  - **2010**: WS, SOA

- **Information Flow**:
  - **Local**
  - **Global**

- **Business Processes**: Otwarte Studium Doktoranckie

- **Web Technologies**:
  - Internet
  - LAN
  - Web
  - XML
  - WS
  - SOA
Seamless Connectivity enables “On Demand” Computing

- Use software as you need
- Much lower setup time, forget about
  - Installation
  - Implementation
  - Training
  - Maintenance
- Scalable and effective usage of resources
  - Provision
  - Billed on true usage
  - Parallelism (CPU, Storage, Bandwidth…)
But Seamless Connectivity is also questioning all our beliefs…

- An application is NOT a single system running on a single device and bounded by a single organization
- Continuum Object … Document
- Messages and Services
  - As opposed « distributed objects »
  - Exchanges becomes peer-to-peer
- Asynchronous communications
- Concurrency becomes the norm while our languages and infrastructures did not plan for it
…we are reaching the point of maximum confusion

- **Federation and Collaboration**
  - As opposed to « Integration »
- **Language(s)**
  - Semantic (not syntactic)
  - Declarative and Model driven (not procedural)
- **Licensing and Deployment models**
- …
So…

• Today, the value is not defined as much by functionality anymore but by connectivity
  – However, we need a new programming model

• Why SOA today?
  – We are reaching a new threshold of connectivity and computing power
From Components to (Web) Services

- Requires a client library
- Client / Server
- Extendable
- Stateless
- Fast
- Small to medium granularity

- Loose coupling via
  - Message exchanges
  - Policies
- Peer-to-peer
- Composable
- Context independent
- Some overhead
- Medium to coarse granularity
Web Services: what is changing?

- **Loose coupling (of course)**
  - Web Services don’t require a CCI (Client side Communication Interface)
    - Very few “gears” needed to consume a service
  - Web Services can accept message content they do not fully understand or support
    - XML, WSDL
  - Web services are very late bound
    - Location is independent of the invocation mechanism
    - Directories
Web Services: What is Changing?

• Peer-to-peer interactions are possible
  – Request / response is an inefficient and very limiting mode of interaction
  – As components coarsen, it is difficult to differentiate a client from a server
Web Services

- loosely coupled, reusable components
- encapsulate discrete functionality
- distributed
- programmatically accessible over standard internet protocols
- add new level of functionality on top of the current web
What Happens to the Technical Services Typically Provided by an Application Server?

- Transaction
- Security
- Connection pooling
- Naming service
- Scalability and failover
- …

- They become the “Service Fabric”
What about the notion of “Container”? They become Service “Domains”

• The notion of “container” shifts to the notion of “Domain Controller”
  – A domain is a collection of web services which share some commonalities like a “secure domain”
  – A container is a domain with one web service
  – Web Services do not always need a container
• Consumers invoke the domain rather than the service directly
• This concept does not exist in any specification…
A Service Fabric can be more than a Bus with a series of Containers / Adapters
The Promise of Web Services

web-based SOA as new system design paradigm

- UDDI Registry
  - Points to Service
  - Points to Description
  - Describes Service
  - Communicates with XML Messages
- WSDL
- SOAP
- Service Consumer
- Web Service
WSDL

- Web Service Description Language
- W3C effort, WSDL 2 final construction phase

describes interface for consuming a Web Service:
- Interface: operations (in- & output)
- Access (protocol binding)
- Endpoint (location of service)
UDDI

- Universal Description, Discovery, and Integration Protocol
- OASIS driven standardization effort

Registry for Web Services:
- provider
- service information
- technical access
SOAP

- Simple Object Access Protocol
- W3C Recommendation

XML data transport:
- sender / receiver
- protocol binding
- communication aspects
- content
Where is WSDL Used?

1. Register web service
2. Search for web service
3. Retrieve WSDL definition
4. Invoke web service
Web Service Invocation
The road to SOA...

Existing business logic, often “model-oriented”

“Global” business logic (tax, trade, …) and data access

Coordination logic (Tx, Process, …) Metadata driven

User Interface NEPs
Shift To A Service-Oriented Architecture

From

• Function oriented
• Build to last
• Prolonged development cycles

To

• Coordination oriented
• Build to change
• Incrementally built and deployed

• Application silos
• Tightly coupled
• Object oriented
• Known implementation

• Enterprise solutions
• Loosely coupled
• Message oriented
• Abstraction

Source: Microsoft (Modified)
So Migrating to SOA

• Would entail
  – Organizing the business logic in a context independent way
    • Typically, model oriented business logic is wrapped behind (web) services

• Re-implementing the controller with “coordination” technologies

• …The view must be completely re-invented
Semantic Web Services Tutorial
Introduction to Semantic Web Services

• The vision of the Semantic Web
• Ontologies as the basic building block
• Current Web Service Technologies
• Vision and Challenges for Semantic Web Services
The Vision

- 500 million users
- more than 3 billion pages

Static

WWW
URI, HTML, HTTP
The Vision

Serious Problems in

- information finding,
- information extracting,
- information representing,
- information interpreting and
- information maintaining.

Static

WWW
URI, HTML, HTTP

Semantic Web
RDF, RDF(S), OWL
The Goal of the Semantic Web

• Fundamental principle:
  Creation and use of semantic metadata

• Using semantics allow:
  better information presentation
  integration information from heterogenous sources
  information classification
The Vision

Dynamic

Web Services
UDDI, WSDL, SOAP

Static

WWW
URI, HTML, HTTP

Semantic Web
RDF, RDF(S), OWL

Bringing the computer back as a device for computation
The Vision

Dynamic

Web Services
UDDI, WSDL, SOAP

Semantic Web Services

Static

WWW
URI, HTML, HTTP

Semantic Web
RDF, RDF(S), OWL

Bringing the web to its full potential
The Semantic Web

• the next generation of the WWW

• information has machine-processable and machine-understandable semantics

• not a separate Web but an augmentation of the current one

• Ontologies as basic building block
Semantic WS

• Semantic description applied to processes

• New WS can be automatically composed by the combination of existing WS
Ontologies and Ontology Languages

An ontology is an explicit and formal specification of a conceptualisation of a domain of interest (Gruber, 1993)

Formal -> permits reasoning by computer
Domain -> a practical knowledge is designed for same particular domain of interest
Ontology Definition

formal, explicit specification of a shared conceptualization

- unambiguous terminology definitions
- machine-readability with computational semantics
- conceptual model of a domain (ontological theory)
- commonly accepted understanding
Ontology definition
(Studer 2004)

O = <C, R, I, A, >
C – set of concepts
R – set of relations (properties)
I – set of instances
A – set of axioms

This definition is used as foundation of OWL
(Web Ontology Language)
Ontology Example

Concept
conceptual entity of the domain

Property
attribute describing a concept

Relation
relationship between concepts or properties

Axiom
cohereency description between Concepts / Properties / Relations via logical expressions

holds(Professor, Lecture) => Lecture.topic = Professor.researchField
Ontology definition (Ehrig 2005)

\[ O = (C, T, R, A, I, V, \leq_C, \leq_T, \sigma_R, \sigma_A, \iota_C, \iota_T, \iota_R, \iota_A) \]

- \( C \) – set of concepts
- \( T \) – set of types
- \( R \) – set of relations
- \( A \) – set of attributes
- \( I \) – set of instances
- \( V \) – set of values
Ontology definition

$\leq_C$ – partial order on $C$ - hierarchy of concepts
$\leq_T$ – partial order on $T$ - hierarchy of types

$\sigma_R : R \rightarrow C \times C$
$\sigma_A : A \rightarrow C \times T$

$\iota_C : C^I$ partial function
$\iota_T : T^V$ partial function

$\iota_R : R \rightarrow 2^{I \times C}$ partial function
$\iota_A : A \rightarrow 2^{I \times V}$ partial function
Representation languages for ontologies

- OIL (Ontology Interchange Language)
- DAML (DARPA Agent Markup Language) + OIL
- OWL (Ontology Web Language)
- KAON (KArlsruhe ONtology)

Based on Resource Description Format (RDF)

- subject-predicate-object statements about resources
- Part of Semantic Web specification
OWL – Web Ontology Language

OWL builds on RDF and RDF Schema and augments the RDFS vocabulary for describing properties and classes:

among others, relations between classes (e.g. disjointedness), cardinality (e.g. “exactly one”), equality, richer typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes.
OWL

OWL provides three increasingly expressive sublanguages designed for use by specific communities of users and implementors:

• **OWL Lite** - which supports users primarily needing a classification hierarchy and simple constraints
• **OWL DL** - which supports users who want maximum expressiveness without losing computational completeness and decidability of reasoning systems.
• **OWL Full** - which is intended for users who want maximum expressiveness and the syntactic freedom of RDF without computational guarantees.
What is RDF?

- Machine-understandable information
- Properties of WWW resources
- Examples: resource discovery, cataloging, content rating, intellectual property rights
- RDF statements specify the properties and values of Web resources
- RDF is encoded in XML
Web resources

• Real resources: anything named by an URL (HTML pages, email addresses, Telnet sessions)
• Proxy resources: electronic representations of Real World things (checking accounts, persons, books, workstations)
Properties

• The relationship between a Web resource and a value (which can be either a string or another resource)
• Properties of a book: author, title, publisher, copyright date, subject, length
• Properties of a bank check: account title, address, date, payee, amount, routing code, account number, check number, memo
## Statements

<table>
<thead>
<tr>
<th>Resource (Subject)</th>
<th>Property (Predicate)</th>
<th>Value (Object)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;War and Peace&quot;</td>
<td>Author</td>
<td>&quot;Leo Tolstoy&quot;</td>
</tr>
<tr>
<td><a href="http://www.ccil.org/~cowan">http://www.ccil.org/~cowan</a></td>
<td>MIME Type</td>
<td>&quot;text/html&quot;</td>
</tr>
</tbody>
</table>
The RDF Format = triples!

- The fundamental design pattern of RDF is to structure your XML data as resource/property/value triples!

```xml
<?xml version="1.0"?>
<Resource-A>
  <property-A>
    <Resource-B>
      <property-B>
        <Resource-C>
          <property-C>
            Value-C
          </property-C>
        </Resource-C>
      </property-B>
    </Resource-B>
  </property-A>
</Resource-A>
```

Notice that the RDF design pattern is an alternating sequence of resource-property. This pattern is known as "striping".
Ontology Technology

To make the Semantic Web working we need:

- **Ontology Languages:**
  - expressivity
  - reasoning support
  - web compliance

- **Ontology Reasoning:**
  - large scale knowledge handling
  - fault-tolerant
  - stable & scalable inference machines

- **Ontology Management Techniques:**
  - editing and browsing
  - storage and retrieval
  - versioning and evolution Support

- **Ontology Integration Techniques:**
  - ontology mapping, alignment, merging
  - semantic interoperability determination

- and ... **Applications**
Deficiencies of WS Technology

• current technologies allow usage of Web Services
• but:
  – only syntactical information descriptions
  – syntactic support for discovery, composition and execution

=> Web Service usability, usage, and integration needs to be inspected manually
  – no semantically marked up content / services
  – no support for the Semantic Web

=> current Web Service Technology Stack failed to realize the promise of Web Services
Semantic Web Services

**Semantic Web Technology**

- allow machine supported data interpretation
- ontologies as data model

**Web Service Technology**

automated discovery, selection, composition, and web-based execution of services

=> Semantic Web Services as integrated solution for realizing the vision of the next generation of the Web
Semantic Web Services

• define exhaustive description frameworks for describing Web Services and related aspects (Web Service Description Ontologies)

• support ontologies as underlying data model to allow machine supported data interpretation (Semantic Web aspect)

• define semantically driven technologies for automation of the Web Service usage process (Web Service aspect)
Semantic Web Services

Usage Process:
• Publication: Make available the description of the capability of a service
• Discovery: Locate different services suitable for a given task
• Selection: Choose the most appropriate services among the available ones
• Composition: Combine services to achieve a goal
• Mediation: Solve mismatches (data, protocol, process) among the combined
• Execution: Invoke services following programmatic conventions
Semantic Web Services

Execution support:

• **Monitoring**: Control the execution process

• **Compensation**: Provide transactional support and undo or mitigate unwanted effects

• **Replacement**: Facilitate the substitution of services by equivalent ones

• **Auditing**: Verify that service execution occurred in the expected way
Semantic Web Services

Initiatives

- WSMO
- OWL-S
- SWSF
  - SWSO
  - SWSL
The Web Service Modeling Ontology WSMO

• Aims & Working Groups
• Design Principles
• Top Level Notions
  – Ontologies
  – Web Services
  – Goals
  – Mediators
• Comparison to OWL-S
WSMO is ..

• a conceptual model for Semantic Web Services:
  – ontology of core elements for Semantic Web Services
  – a formal description language (WSML)
  – execution environment (WSMX)

• derived from and based on the Web Service Modeling Framework WSMF

• a SDK-Cluster Working Group
  (joint European research and development initiative)
WSMO Working Groups

A Conceptual Model for SWS

A Formal Language for WSMO
A Rule-based Language for SWS

SdK

Execution Environment for WSMO
WSMO Design Principles

- Web Compliance
- Ontology-Based
- Goal-driven
- Strict Decoupling
- Centrality of Mediation
- Description versus Implementation
- Execution Semantics
WSMO Top Level Notions

Objectives that a client wants to achieve by using Web Services

Provide the formally specified terminology of the information used by all other components

Semantic description of Web Services:
- **Capability** (functional)
- **Interfaces** (usage)

Connectors between components with mediation facilities for handling heterogeneities

**WSMO D2, version 1.2, 13 April 2005 (W3C submission)**
Non-Functional Properties

every WSMO elements is described by properties that contain relevant, non-functional aspects

- Dublin Core Metadata Set:
  - complete item description
  - used for resource management
- Versioning Information
  - evolution support
- Quality of Service Information
  - availability, stability
- Other
  - Owner, financial
## Non-Functional Properties List

### Dublin Core Metadata
- Contributor
- Coverage
- Creator
- Description
- Format
- Identifier
- Language
- Publisher
- Relation
- Rights
- Source
- Subject
- Title
- Type

### Quality of Service
- Accuracy
- NetworkRelatedQoS
- Performance
- Reliability
- Robustness
- Scalability
- Security
- Transactional
- Trust

### Other
- Financial
- Owner
-.TypeOfMatch
- Version
WSMO Ontologies

Objectives that a client wants to achieve by using Web Services

- Provide the formally specified terminology of the information used by all other components
- Semantic description of Web Services:
  - Capability (functional)
  - Interfaces (usage)

Connectors between components with mediation facilities for handling heterogeneities
Ontology Usage & Principles

- **Ontologies are used as the ‘data model’ throughout WSMO**
  - all WSMO element descriptions rely on ontologies
  - all data interchanged in Web Service usage are ontologies
  - Semantic information processing & ontology reasoning

- **WSMO Ontology Language WSML**
  - conceptual syntax for describing WSMO elements
  - logical language for axiomatic expressions (WSML Layering)

- **WSMO Ontology Design**
  - **Modularization:** import / re-using ontologies, modular approach for ontology design
  - **De-Coupling:** heterogeneity handled by **OO Mediators**
Ontology Specification

• **Non functional properties** (see before)
• **Imported Ontologies** importing existing ontologies where no heterogeneities arise
• **Used mediators** OO Mediators (ontology import with terminology mismatch handling)

**Ontology Elements:**

- **Concepts** set of concepts that belong to the ontology, incl.
- **Attributes** set of attributes that belong to a concept
- **Relations** define interrelations between several concepts
- **Functions** special type of relation (unary range = return value)
- **Instances** set of instances that belong to the represented ontology
- **Axioms** axiomatic expressions in ontology (logical statement)
WSMO Web Services

Objectives that a client wants to achieve by using Web Services

Provide the formally specified terminology of the information used by all other components

Connectors between components with mediation facilities for handling heterogeneities

Semantic description of Web Services:
- **Capability** (functional)
- **Interfaces** (usage)
WSMO Web Service Description

Non-functional Properties
- complete item description
- quality aspects
- Web Service Management

Non-functional Properties
- Advertising of Web Service
- Support for WS Discovery

Capability
- functional description

Web Service Implementation
(not of interest in Web Service Description)

Choreography --- Service Interfaces --- Orchestration

- client-service interaction interface for consuming WS
  - External Visible Behavior
  - Communication Structure
  - ‘Grounding’

DC + QoS + Version + financial

realization of functionality by aggregating other Web Services
- functional decomposition
- WS composition

© DSRG 2006
www.cs.agh.edu.pl
Capability Specification

- **Non functional properties**
- **Imported Ontologies**
- **Used mediators**
  - *OO Mediator*: importing ontologies with mismatch resolution
  - *WG Mediator*: link to a Goal wherefore service is not usable a priori
- **Pre-conditions**
  What a web service expects in order to be able to provide its service. They define conditions over the input.
- **Assumptions**
  Conditions on the state of the world that has to hold before the Web Service can be executed
- **Post-conditions**
  describes the result of the Web Service in relation to the input, and conditions on it
- **Effects**
  Conditions on the state of the world that hold after execution of the Web Service (i.e. changes in the state of the world)
Choreography & Orchestration

- **VTA example:**

  ![Diagram of VTA example]

  - **Choreography** = how to interact with the service to consume its functionality
  - **Orchestration** = how service functionality is achieved by aggregating other Web Services
Choreography Aspects

*Interface for consuming Web Service*

- **External Visible Behavior**
  - those aspects of the workflow of a Web Service where Interaction is required
  - described by workflow constructs: sequence, split, loop, parallel

- **Communication Structure**
  - messages sent and received
  - their order (communicative behavior for service consumption)

- **Grounding**
  - executable communication technology for interaction
  - choreography related errors (e.g. input wrong, message timeout, etc.)

- **Formal Model**
  - reasoning on Web Service interfaces (service interoperability)
  - allow mediation support on Web Service interfaces
Orchestration Aspects

Control Structure for aggregation of other Web Services

- decomposition of service functionality
- all service interaction via choreographies
WSMO Web Service Interfaces

- service interfaces are concerned with service consumption and interaction
- Choreography and Orchestration as sub-concepts of Service Interface
- common requirements for service interface description:
  1. represent the dynamics of information interchange during service consumption and interaction
  2. support ontologies as the underlying data model
  3. appropriate communication technology for information interchange
  4. sound formal model / semantics of service interface specifications in order to allow operations on them.
Service Interface Description

• **Ontologies as data model:**
  – all data elements interchanged are ontology instances
  – service interface = evolving ontology

• **Abstract State Machines (ASM) as formal framework:**
  – dynamics representation: high expressiveness & low ontological commitment
  – core principles: state-based, state definition by formal algebra, guarded transitions for state changes
  – overcome the “Frame Problem”

• **further characteristics:**
  – not restricted to any specific communication technology
  – ontology reasoning for service interoperability determination
  – basis for declarative mediation techniques on service interfaces
Future Directions

**Choreography:**
- interaction of services / service and client
- a „choreography interface” describes the behavior of a Web Service for client-service interaction for consuming the service

**Orchestration:**
- how the functionality of a Web Service is achieved by aggregating other Web Services
- extends Choreography descriptions by control & data flow constructs between orchestrating WS and orchestrated WSs.

**Conceptual models**

**User language**
- based on UML2 activity diagrams
- graphical Tool for Editing & Browsing Service Interface Description

**Workflow constructs as basis for describing service interfaces:**
- workflow based process models for describing behavior
- on basis of generic workflow constructs (e.g. van der Aalst)

**Formal description of service interfaces:**
- ASM-based approach
- allows reasoning & mediation

**Ontologies as data model:**
- every resource description based on ontologies
- every data element interchanged is ontology instance

**Grounding:**
- making service interfaces executable
- currently grounding to WSDL
WSMO Goals

Objectives that a client wants to achieve by using Web Services

Provide the formally specified terminology of the information used by all other components

Semantic description of Web Services:
- **Capability** *(functional)*
- **Interfaces** *(usage)*

Connectors between components with mediation facilities for handling heterogeneities
Goals

• Ontological De-coupling of Requester and Provider

• **Goal-driven Approach**, derived from AI rational agent approach
  - requester formulates objective independently
  - ‘intelligent’ mechanisms detect suitable services for solving the Goal
  - allows re-use of Services for different purposes

• **Usage of Goals within Semantic Web Services**
  – A requester (human or machine) defines a Goal to be resolved
  – Web Service discovery detects suitable Web Services for solving the Goal automatically
  – Goal resolution management is realized in implementations
Goal Specification

- **Non functional properties**
- **Imported Ontologies**
- **Used mediators**
  - *OO Mediators*: importing ontologies with heterogeneity resolution
  - *GG Mediator*:
    - Goal definition by reusing an already existing goal
    - allows definition of *Goal Ontologies*
- **Requested Capability**
  - describes service functionality expected to resolve the objective
  - defined as capability description from the requester perspective
- **Requested Interface**
  - describes communication behaviour supported by the requester for consuming a Web Service (Choreography)
  - Restrictions / preferences on orchestrations of acceptable Web Services
WSMO Mediators

Objectives that a client wants to achieve by using Web Services

Provide the formally specified terminology of the information used by all other components

Semantic description of Web Services:
- Capability (functional)
- Interfaces (usage)

Connectors between components with mediation facilities for handling heterogeneities
Mediation

• **Heterogeneity …**
  – Mismatches on structural / semantic / conceptual / level
  – Occur between different components that shall interoperate
  – Especially in distributed & open environments like the Internet

• **Concept of Mediation** (Wiederhold, 94):
  – *Mediators* as components that resolve mismatches
  – **Declarative Approach:**
    • Semantic description of resources
    • ‘Intelligent’ mechanisms that resolve mismatches independent of content
  – Mediation cannot be fully automated (integration decision)

• **Levels of Mediation within Semantic Web Services** (WSMF):
  1. **Data Level:** mediate heterogeneous **Data Sources**
  2. **Protocol Level:** mediate heterogeneous **Communication Patterns**
  3. **Process Level:** mediate heterogeneous **Business Processes**
WSMO Mediators Overview
Mediator Structure

WSMO Mediator uses a Mediation Service via

- as a Goal
- directly
- optionally incl. Mediation
OO Mediator - Example

Merging 2 ontologies

Goal:
“merge s1, s2 and s1.ticket subclassof s2.product”

Discovery

Mediation Services

Train Connection Ontology (s1)

Purchase Ontology (s2)

OO Mediator Mediation Service

Train Ticket Purchase Ontology
GG Mediators

- **Aim:**
  - Support specification of Goals by re-using existing Goals
  - Allow definition of **Goal Ontologies** (collection of pre-defined Goals)
  - Terminology mismatches handled by OO Mediators

- **Example: Goal Refinement**

  ![Diagram](source-goal-buy-a-ticket-gg-mediator-target-goal-buy-a-train-ticket-postcondition-a-ticket-memberof-trainticket)
WG & WW Mediators

• **WG Mediators:**
  - link a Web Service to a Goal and resolve occurring mismatches
  - match Web Service and Goals that do not match a priori
  - handle terminology mismatches between Web Services and Goals
  ⇒ broader range of Goals solvable by a Web Service

• **WW Mediators:**
  - enable interoperability of heterogeneous Web Services
  ⇒ support automated collaboration between Web Services

  – **OO Mediators** for terminology import with data level mediation
  – Protocol Mediation for establishing valid multi-party collaborations
  – Process Mediation for making Business Processes interoperable
Comparison to OWL-S

- Mapping to WSDL
  - communication protocol (RPC, HTTP, ...)
  - marshalling/serialization
  - transformation to and from XSD to OWL

- Control flow of the service
  - Black/Grey/Glass Box view
  - Protocol Specification
  - Abstract Messages

- Capability specification
- General features of the Service
  - Quality of Service
  - Classification in Service taxonomies
Perspective

- **OWL-S** is an ontology and a language to describe Web services
  - Strong relation to Web Services standards
    - rather than proposing another WS standard, OWL-S aims at enriching existing standards
    - OWL-S is grounded in WSDL and it has been mapped into UDDI
  - Based on the Semantic Web
    - Ontologies provide conceptual framework to describe the domain of Web services and an inference engine to reason about the domain
    - Ontologies are essential elements of interoperability between Web services

- **WSMO** is a conceptual model for the core elements of Semantic Web Services
  - core elements: Ontologies, Web Services, Goals, Mediators
    - language for semantic element description (WSML)
    - reference implementation (WSMX)
  - Mediation as a key element
  - Ontologies as data model
    - every resource description is based on ontologies
    - every data element interchanged is an ontology instance
OWL-S and WSMO

OWL-S profile $\approx$ WSMO capability + goal + non-functional properties

- OWL-S uses Profiles to express existing capabilities (advertisements) and desired capabilities (requests)
- WSMO separates provider (capabilities) and requester points of view (goals)
OWL-S and WSMO

OWL-S Process Model ≈ WSMO Service Interfaces

• Perspective:
  – OWL-S Process Model describes operations performed by Web Service, including consumption as well as aggregation
  – WSMO separates Choreography and Orchestration

• Formal Model:
  – OWL-S formal semantics has been developed in very different frameworks such as Situation Calculus, Petri Nets, Pi-calculus
  – WSMO service interface description model with ASM-based formal semantics
  – OWL-S Process Model is extended by SWRL / FLOWS

both approaches are not finalized yet
OWL-S and WSMO

OWL-S Grounding ≈ current WSMO Grounding

• OWL-S provides default mapping to WSDL
  – clear separation between WS description and interface implementation
  – other mappings could be used

• WSMO also defines a mapping to WSDL, but aims at an ontology-based grounding
  – avoid loss of ontological descriptions throughout service usage process
  – ‘Triple-Spaced Computing’ as innovative communication technology
Mediation in OWL-S and WSMO

- OWL-S does not have an explicit notion of mediator
  - Mediation is a by-product of the orchestration process
    - E.g. protocol mismatches are resolved by constructing a plan that coordinates the activity of the Web services
    - …or it results from translation axioms that are available to the Web services
    - It is not the mission of OWL-S to generate these axioms
- WSMO regards mediators as key conceptual elements
  - Different kinds of mediators:
    - OO Mediators for ensuring semantic interoperability
    - GG, WG mediators to link Goals and Web Services
    - WW Mediators to establish service interoperability
  - Reusable mediators
  - Mediation techniques under development
Semantic Representation

- **OWL-S** and **WSMO** adopt a similar view on the need of ontologies and explicit semantics but they rely on different logics:
  - **OWL-S** is based on **OWL / SWRL**
    - OWL represents taxonomical knowledge
    - SWRL provides inference rules
    - FLOWS as formal model for process model
  - **WSMO** is based on **WSML** a family of languages with a common basis for compatibility and extensions in the direction of Description Logics and Logic Programming
• WSML aims at overcoming deficiencies of OWL
• Relation between WSML and OWL+SWRL to be completed
# Summary

<table>
<thead>
<tr>
<th></th>
<th>OWL-S</th>
<th>WSMO</th>
<th>current Web Service technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Discovery</strong></td>
<td>Profile</td>
<td>Goals and Web Services (capability)</td>
<td></td>
</tr>
<tr>
<td>detection of suitable WS</td>
<td></td>
<td></td>
<td><strong>UDDI API</strong></td>
</tr>
<tr>
<td><strong>Consumption &amp; Interaction</strong></td>
<td>Process Model</td>
<td>Service Interfaces (Choreography + Orchestration)</td>
<td></td>
</tr>
<tr>
<td>How to consume &amp; aggregate</td>
<td></td>
<td></td>
<td><strong>BPEL4WS / WS-CDL</strong></td>
</tr>
<tr>
<td><strong>Invocation</strong></td>
<td>Grounding+WSDL/SOAP</td>
<td>Grounding (WSDL / SOAP, ontology-based)</td>
<td></td>
</tr>
<tr>
<td>How to invoke</td>
<td></td>
<td></td>
<td><strong>WSDL / SOAP</strong></td>
</tr>
<tr>
<td><strong>Mediation</strong></td>
<td>-</td>
<td>Mediators</td>
<td></td>
</tr>
<tr>
<td>Heterogeneity handling</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
End of lecture 3