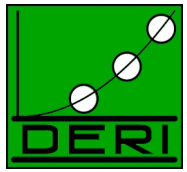


BIT-Seminar, 16/03/2005, Bolzano

Current Efforts towards Semantic Web Services (SWS): OWL-S and WSMO

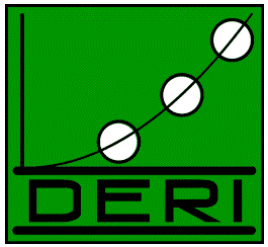
Axel Polleres axel.polleres@deri.org

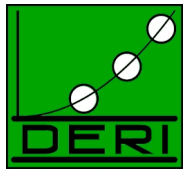
Slides partly based on recent Tutorial at ISWC'04 (Hiroshima) by:
Sinuhe Arroyo, Christoph Bussler, Jos de Brujin, Ruben Lara, David Martin (OWL-S), Matthew Moran, Massimo Paolucci (OWL-S), Michael Stollberg, Katia Sycara (OWL-S), Michal Zaremba, Laurentiu Vasiliu, Liliana Cabral, John Domingue



Semantic Web Services:

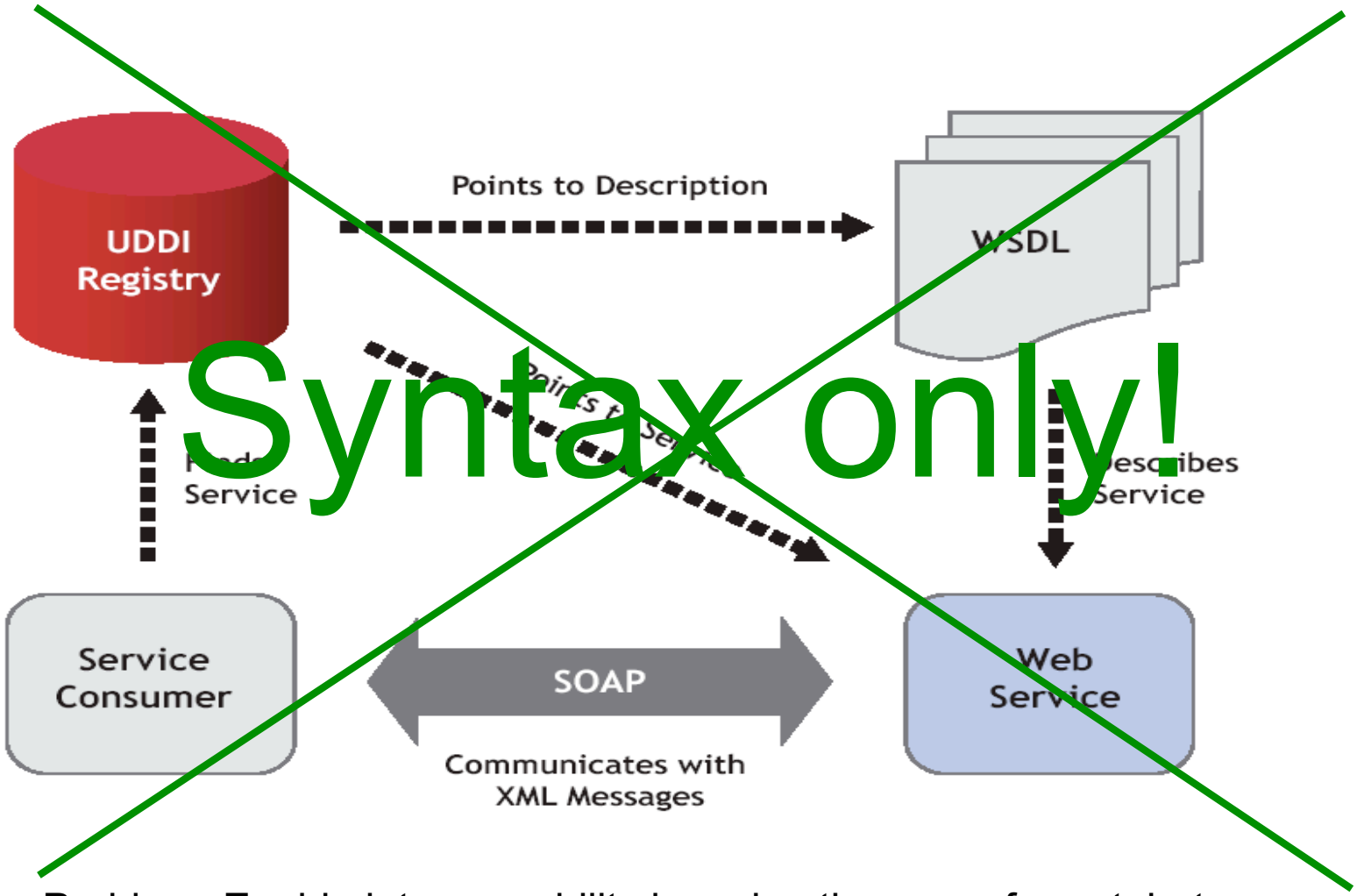
- **Introduction to Semantic Web Services (SWS)**
- OWL-S & WSMO
- Comparison



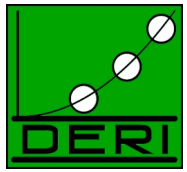


Semantic Web Services
=
Semantic Web Technology
+
Web Service Technology

WS standards: Lack of semantics!



Problem: Enable interoperability by using the same format, but still discovery, combinability only syntax based, no way to describe functionality.



Semantic Web Services (2)

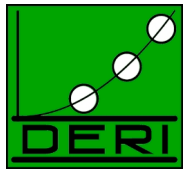
Semantic Web:

- Ontologies - basic building block:
  "Formal, explicit specification of a shared conzeptualization"
- Allow machine supported data interpretation
- Ontology Language Standards:
  - RDF, RDFS … triples, graph based model
  - OWL … DL (+extensions SWRL, full FOL)
  - WSML … LP, F-Logic, …

i.e.

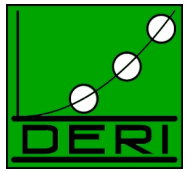
- instance data, plus relations between instances (RDF)
- modeling taxonomies (RDFS)
- richer inference rules and axioms over my instances and relations (OWL, OWL-S, F-Logic, SWRL, WSML)

Semantic annotation shall enable machine-processable data and automation of processing the data on the Web!



Semantic Web Services

- What should S+WS and service ontologies provide?
  (Partly) Automation of the 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
  - Monitoring: Control the execution process
  - Compensation: Provide transactional support and undo or mitigate unwanted effects
  - Replacement: Facilitate the substitution of services by equivalent ones



Service Description languages and Ontologies to enable semantic markup

- Should describe all information necessary to enable automating discovery, composition, execution, etc.
- Semantically enhanced repositories
- Tools and platforms that:
  - semantically enrich current Web content
  - facilitate discovery, composition and execution

Two main efforts: **OWL-S** and **WSMO**



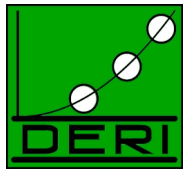
Semantic Web Services:

- Introduction to Semantic Web Services (SWS)
- **OWL-S & WSMO**
- OWL-S and WSMO - Design decisions and trade-offs

Intelligent Software Agents
Carnegie Mellon University

DERI

KNOWLEDGE MEDIA
KMi
INSTITUTE



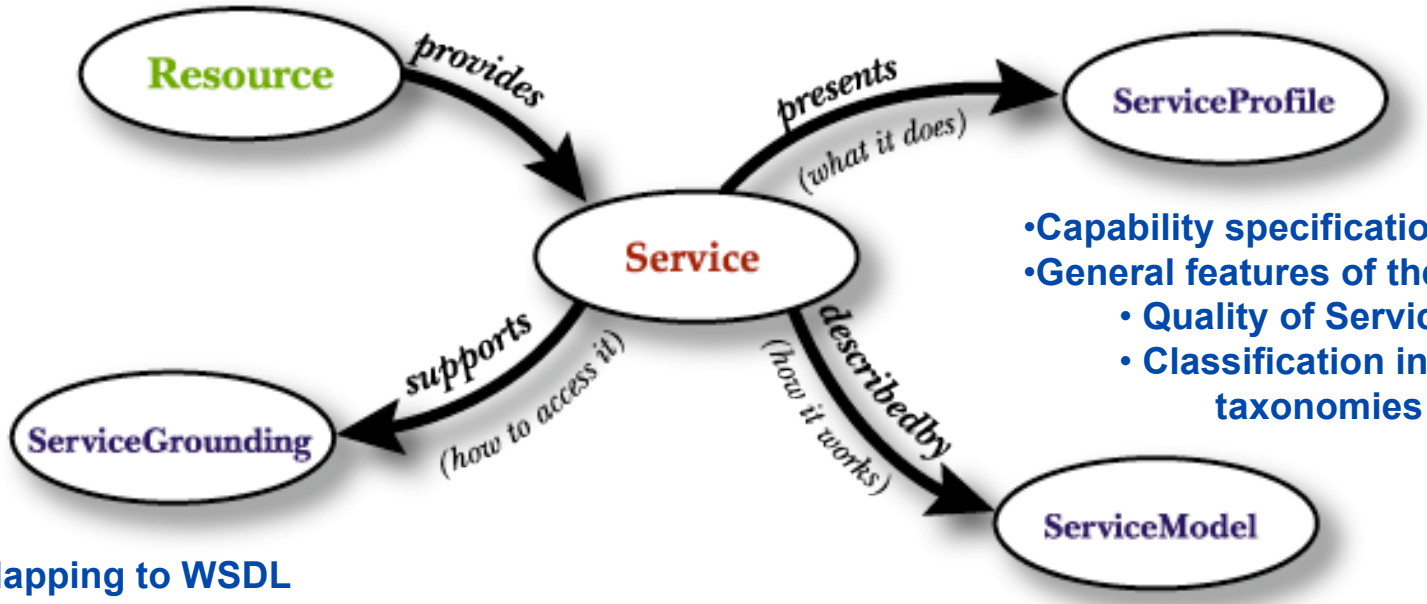
OWL-S Ontology

- OWL-S is an OWL ontology to describe Web services
- OWL-S leverages on OWL to
  - Support capability based discovery of Web services
  - Support automatic composition of Web Services
  - Support automatic invocation of Web services

"Complete do not compete"

  - OWL-S does not aim to replace the Web services standards rather OWL-S attempts to provide a semantic layer
    - OWL-S relies on WSDL for Web service invocation *(see Grounding)*
    - OWL-s Expands UDDI for Web service discovery *(OWL-S/UDDI mapping)*

OWL-S Upper Ontology

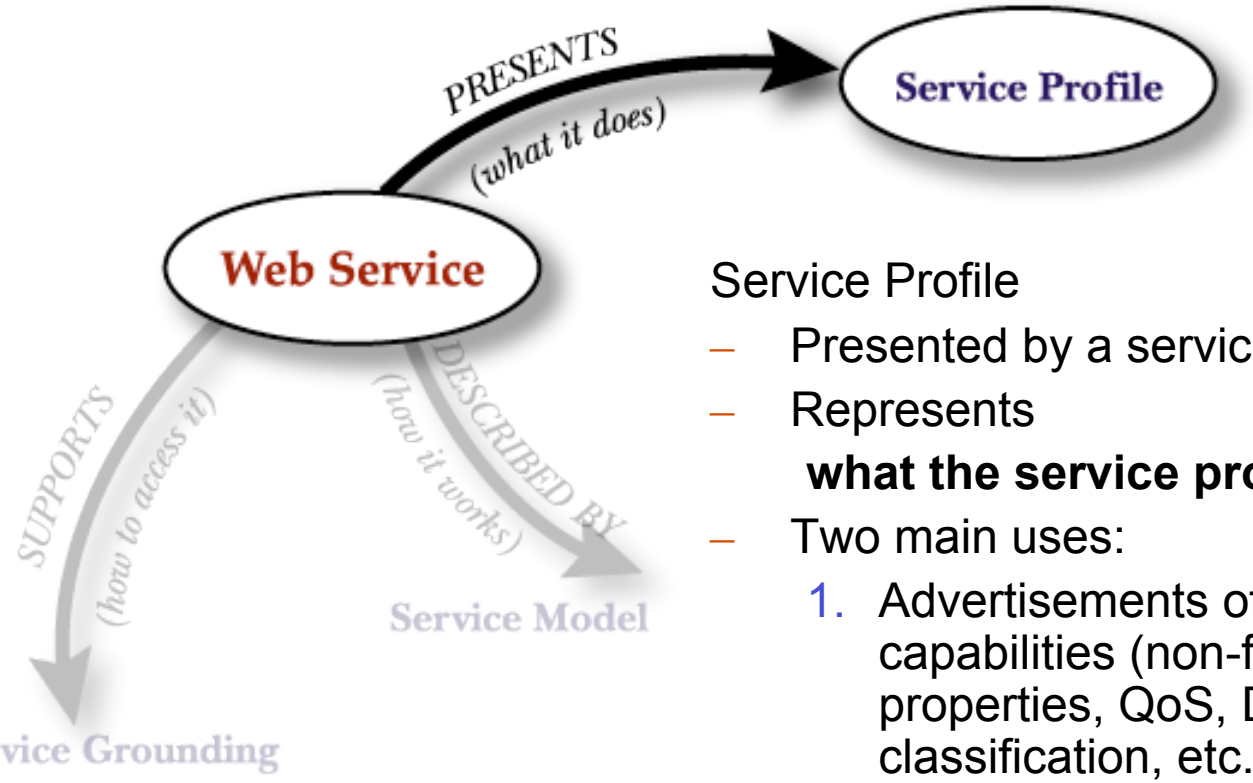


- Capability specification
- General features of the Service
  - Quality of Service
  - Classification in Service taxonomies

- 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

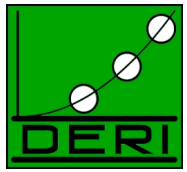
Service Profiles



Service Profile

- Presented by a service.
- Represents
  what the service provides
- Two main uses:
  1. Advertisements of Web Services capabilities (non-functional properties, QoS, Description, classification, etc.)
  2. Request of Web services with a given set of capabilities

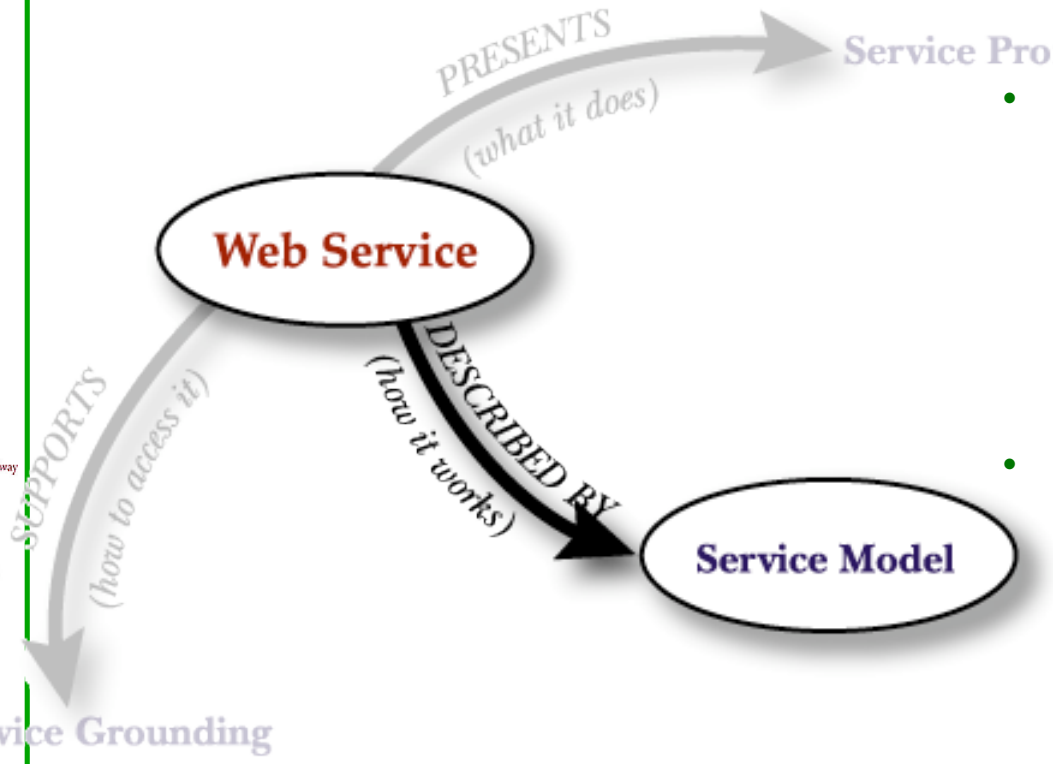
•**Profile does not specify use/invocation!**



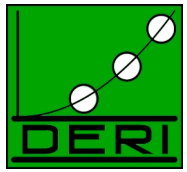
OWL-S Service Profile
Capability Description

- **Preconditions**
  - Set of conditions that should hold prior to service invocation
- **Inputs**
  - Set of necessary inputs that the requester should provide to invoke the service
- **Outputs**
  - Results that the requester should expect after interaction with the service provider is completed
- **Effects**
  - Set of statements that should hold true if the service is invoked successfully.
- **Service type**
  - What kind of service is provided (eg selling vs distribution)
- **Product**
  - Product associated with the service (eg travel vs books vs auto parts)

Process Model



- Process Model
  - Describes how a service works: internal processes of the service
  - Specifies service interaction protocol
  - Specifies abstract messages: ontological type of information transmitted
- Facilitates
  - Web service invocation
  - Composition of Web services
  - Monitoring of interaction



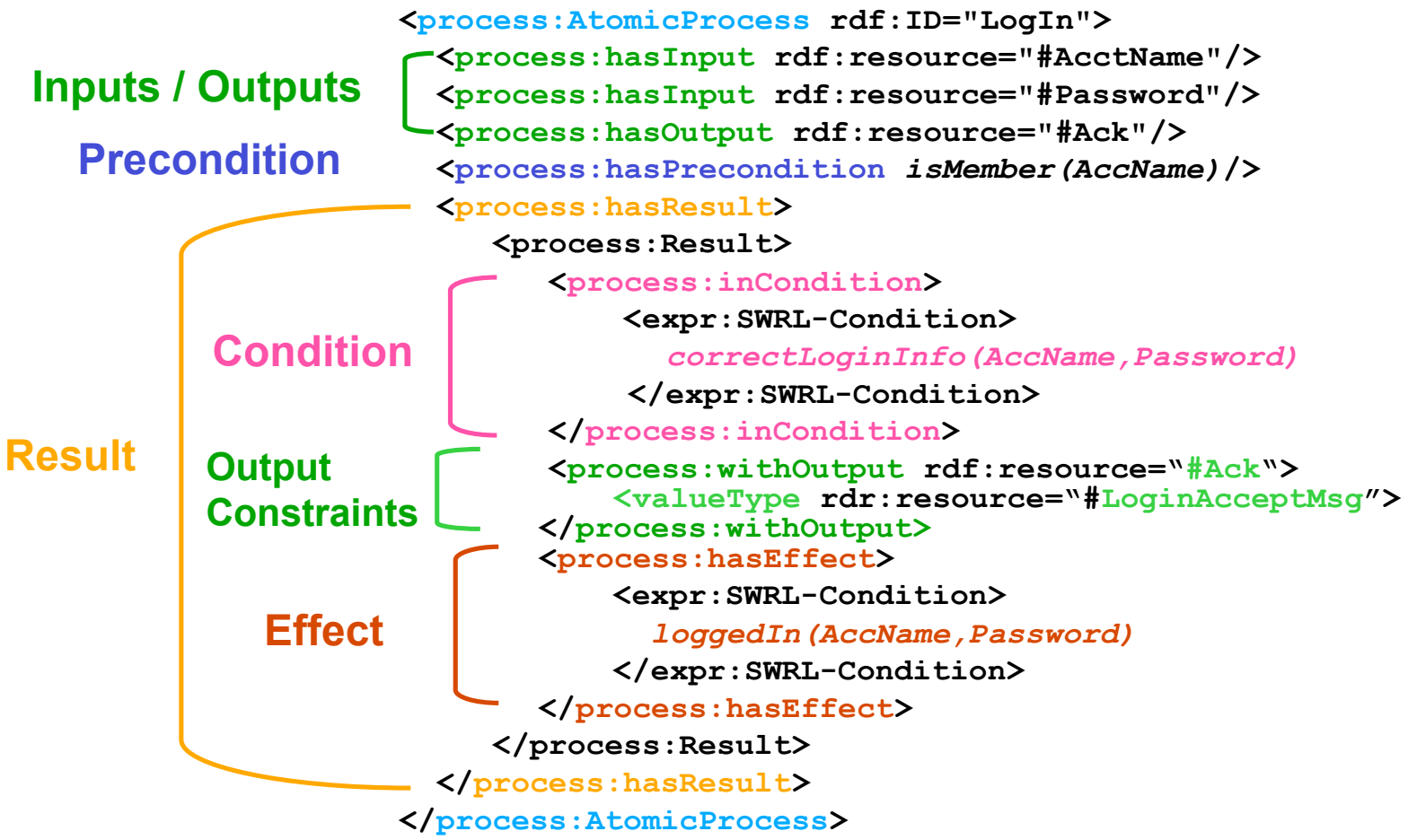
Definition of Process

- A Process represents a transformation (function).

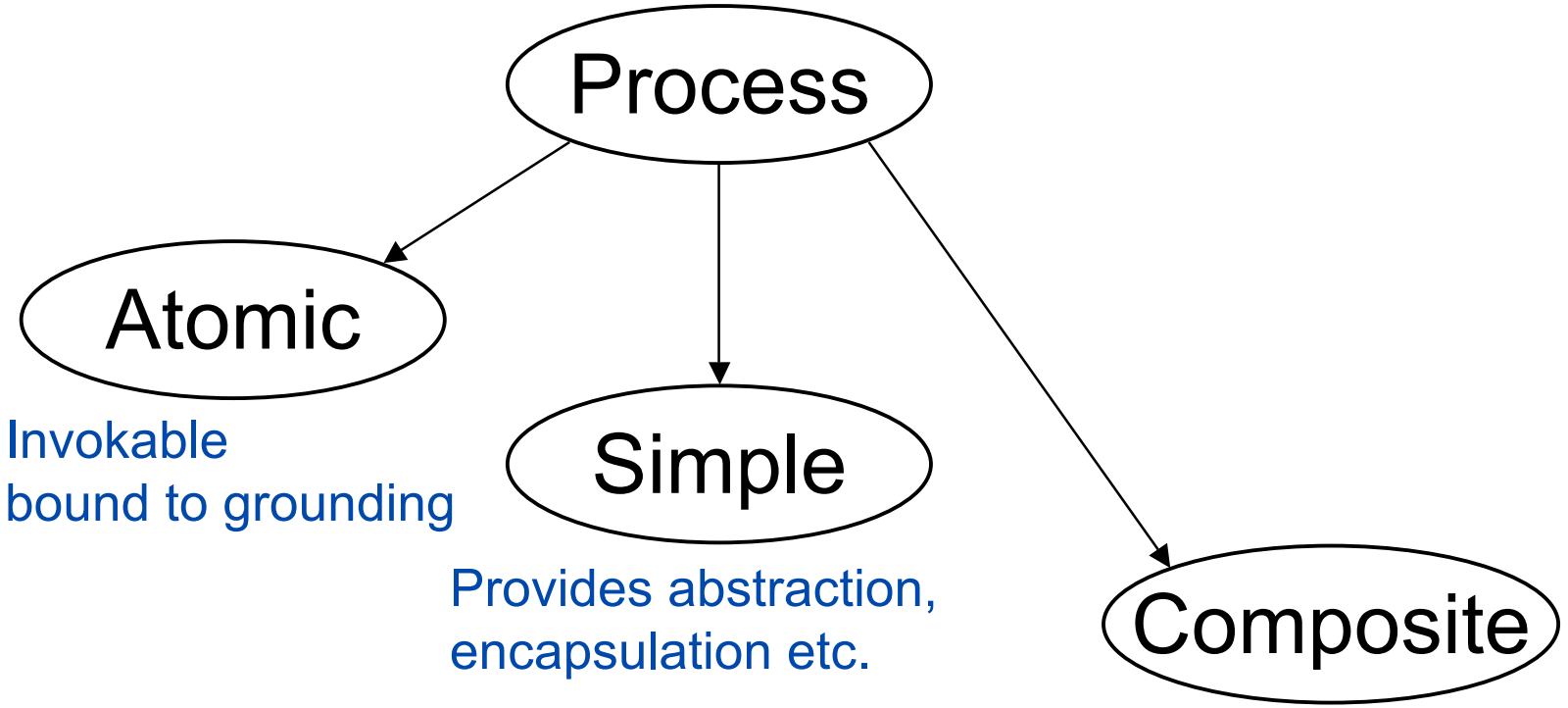
It is characterized by four parameters

- ***Inputs***: the inputs that the process requires
- ***Preconditions***: the conditions that are required for the process to run correctly
- ***Outputs***: the information that results from (and is returned from) the execution of the process
- ***Results***: a process may have different outcomes depending on some condition
  - ***Condition***: under what condition the result occurs
  - ***Constraints on Outputs***
  - ***Effects***: real world changes resulting from the execution of the process

Example of an atomic Process



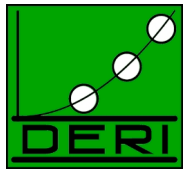
Ontology of Processes



Invokable
bound to grounding

Provides abstraction,
encapsulation etc.

Defines a workflow
composed of process
performs



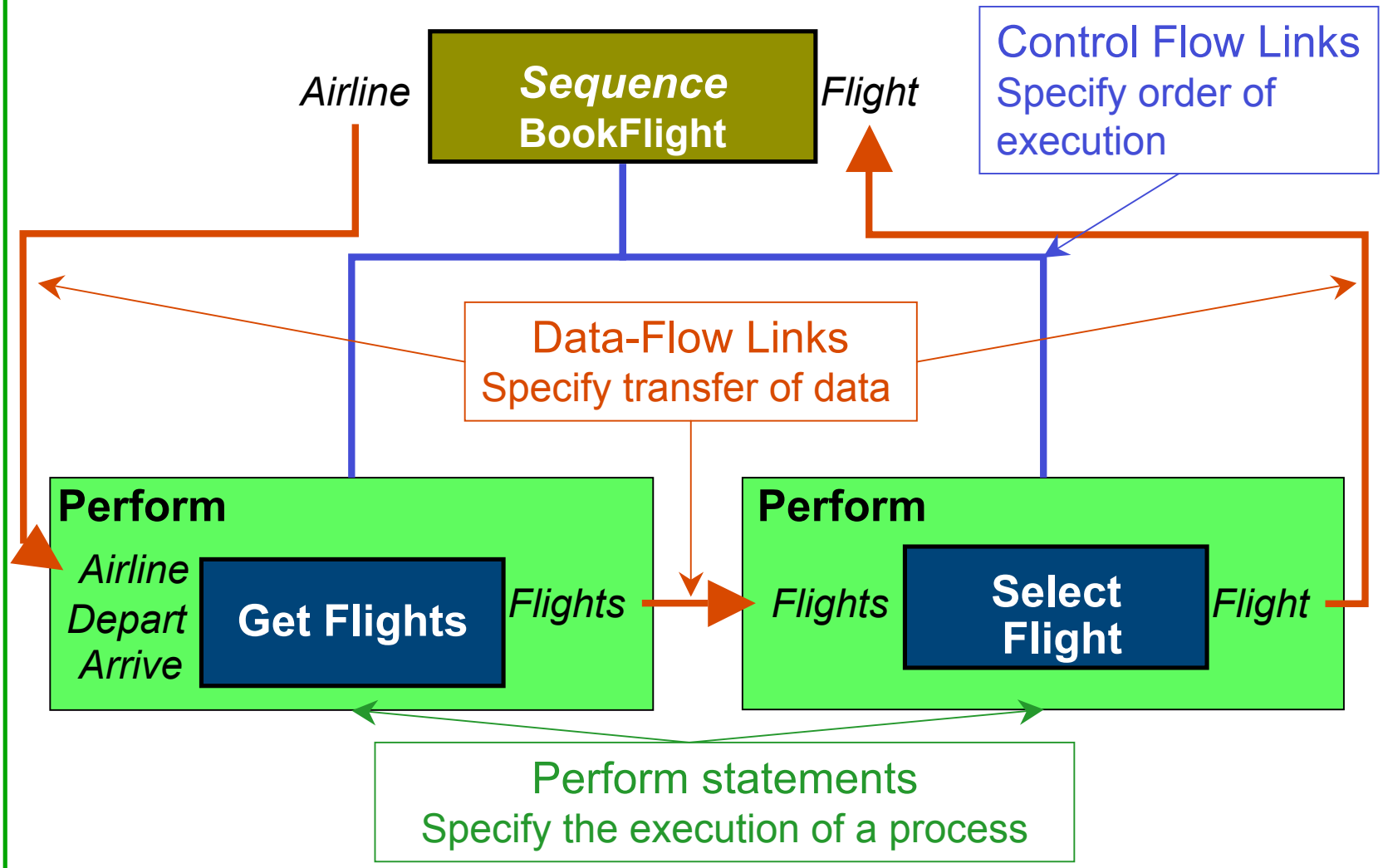
Composite Processes

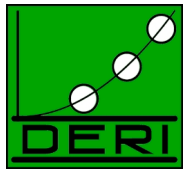
- Composite Processes specify how processes work together to compute a complex function
- Composite processes define
  1. **Control Flow**

     Specify the temporal relations between the executions of the different sub-processes (sequence, choice, etc.)
  2. **Data Flow**

     Specify how the data produced by one process is transferred to another process

Example of Composite Process



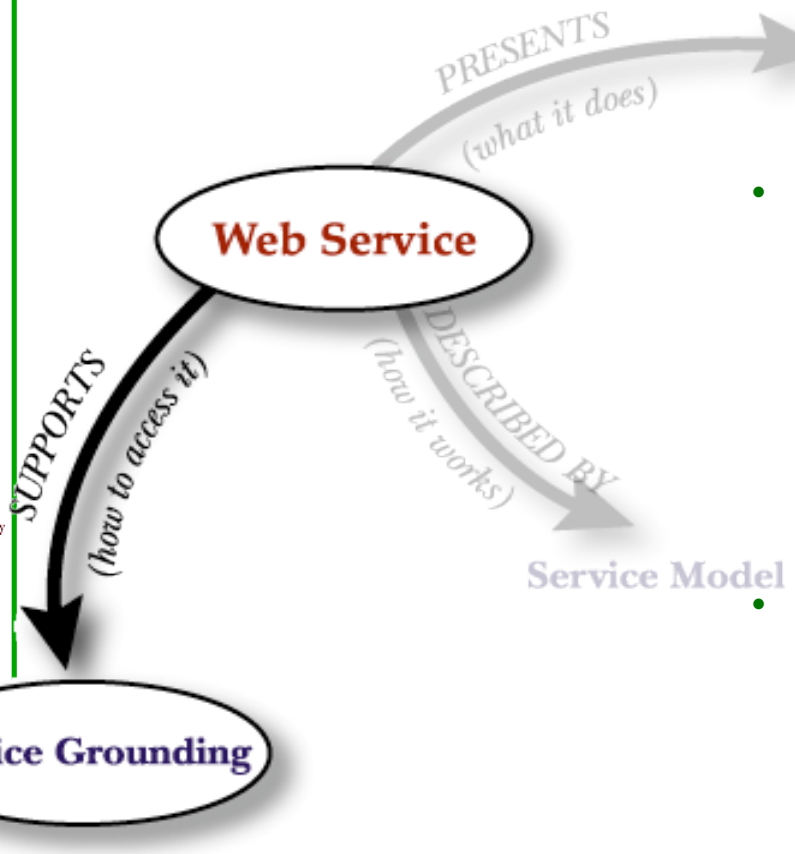


Process Model Organization

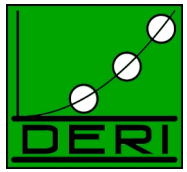
- **Process Model is described as a tree structure**
  - Composite processes are internal nodes
  - Simple and Atomic Processes are the leaves
- **Simple processes represent an abstraction**
  - Placeholders of processes that aren't specified
  - Or that may be expressed in many different ways
- **Atomic Processes correspond to the basic actions that the Web service performs**
  - Hide the details of how the process is implemented
  - Correspond to WSDL operations

~ related Process Definition Languages a la BPEL

Service Grounding

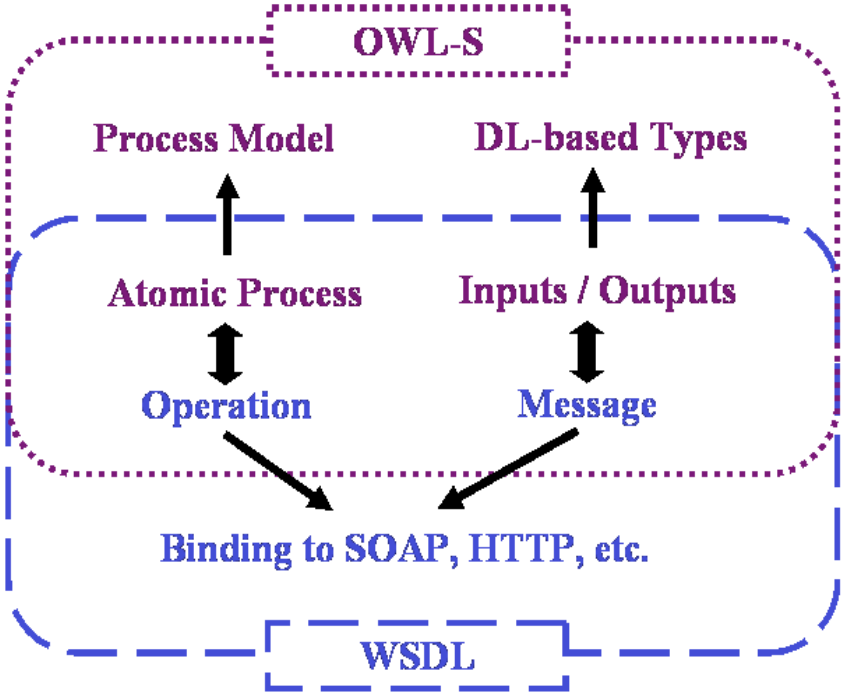


- Service Grounding
  - Provides a specification of service access information.
  - Service Model + Grounding give everything needed for using the service
  - Builds upon **WSDL** to define message structure and physical binding layer
- Specifies:
  - communication protocols, transport mechanisms, communication languages, etc.

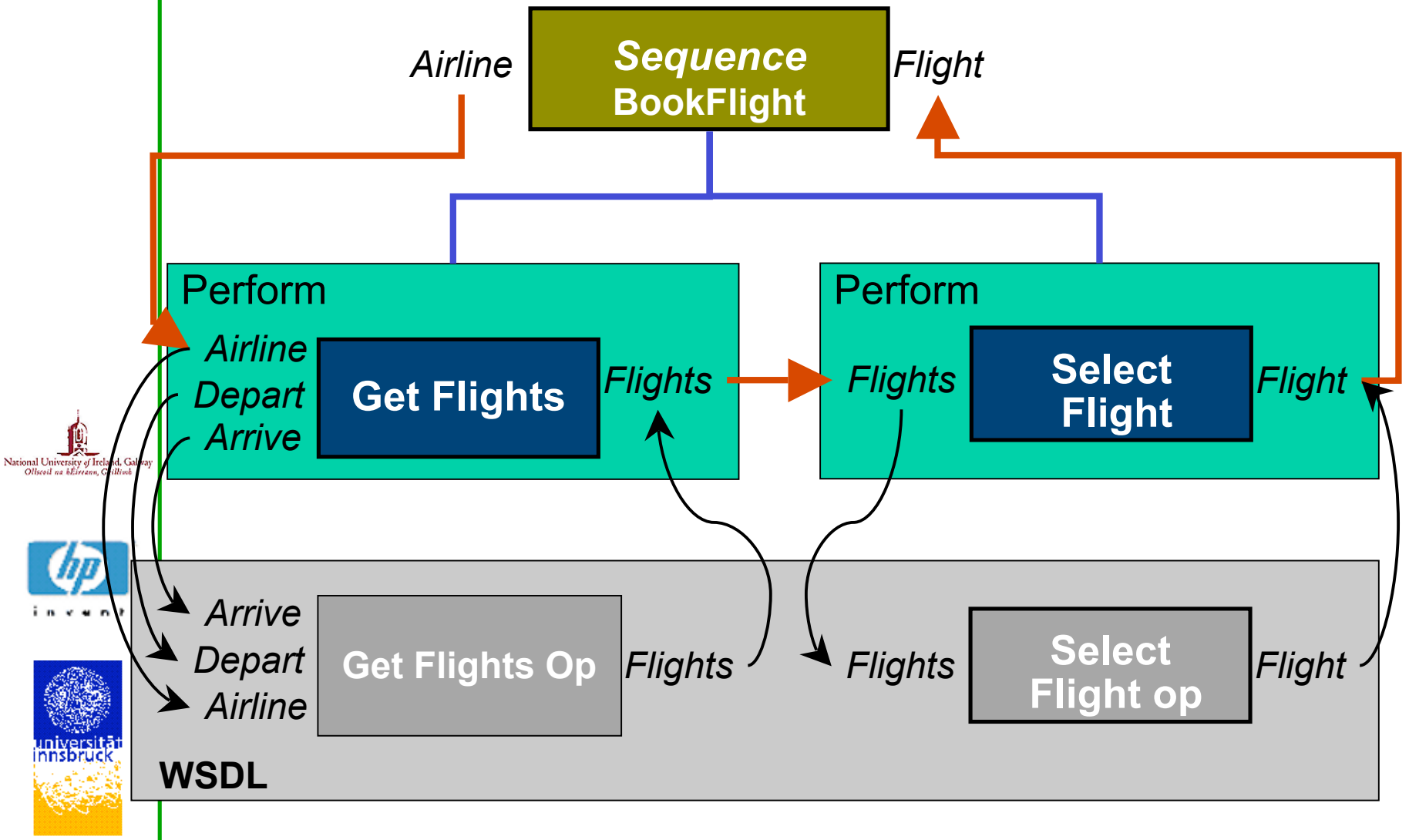


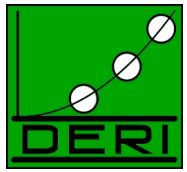
Mapping OWL-S / WSDL 1.1

- **Operations** correspond to Atomic Processes
- **Input/Output** messages correspond to Inputs/Outputs of processes



Example of Grounding

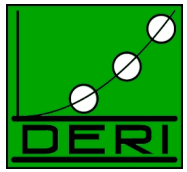




Result of using the Grounding

- Invocation mechanism for OWL-S
  - Invocation based on WSDL
  - Different types of invocation supported by WSDL can be used with OWL-S
- Clear separation between service description and invocation/implementation
  - Service description is needed to reason about the service
    - Decide how to use it
    - Decide how what information to send and what to expect
  - Service implementation may be based on SOAP an XSD types
  - The crucial point is that the information that travels on the wires and the information used in the ontologies is the same
- Allows any web service to be represented using OWL-S

Personal Remark: I do not completely believe this enables composition: still different SOAP messages can be linked to the same service: ambiguities!



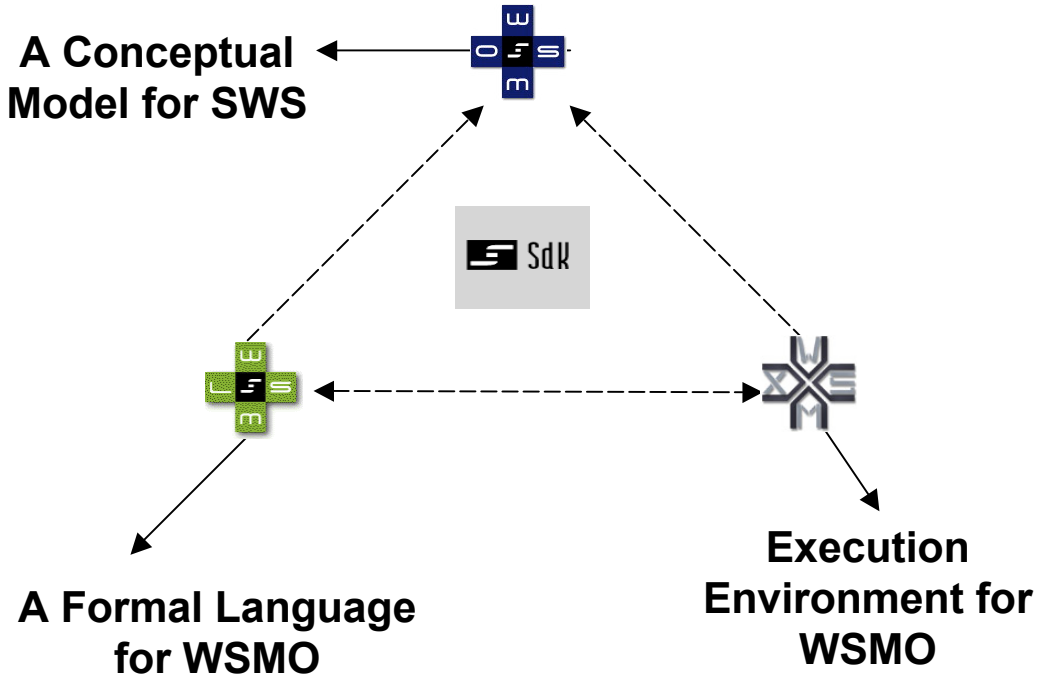
OWL-S: Language
Some superficial comments:

- OWL-S itself is an OWL Ontology,
- Combined with SWRL for preconditions and effects.
- Inputs/Outputs subclasses of SWRL variables
- Possible candidates for logicical language used: SWRL, SWRL-FOL, (KIF, DRS)

- However: Dicsovery, composition approaches published so far operate purely on description logic reasoning

WSMO

- WSMO is an ontology and conceptual framework to describe Web services and related aspects
- Based Web Service Modeling Framework (WSMF)
- WSMO is a SDK-Cluster Working Group

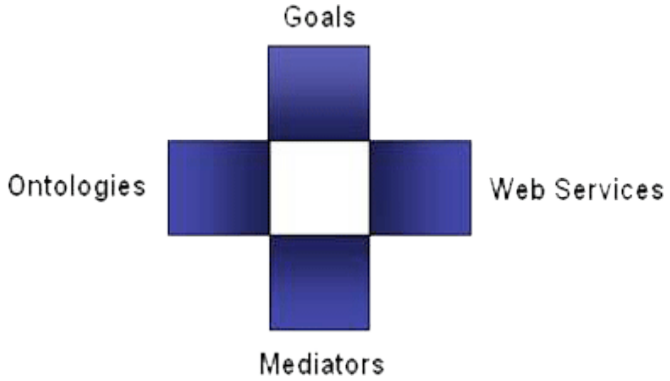


WSMO Principles and Top Level Concepts:

- **Strong Decoupling & Strong Mediation**
  - *autonomous components with mediators for interoperability*
- **Interface vs. Implementation:**
  - *distinguish interface (= description) from implementation (=program)*

Objectives that a client may have when consulting a Web Service

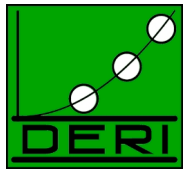
Provide the formally specified terminology of the information used by all other components



Semantic description of Web Services

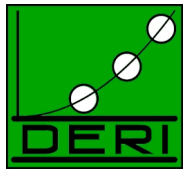
Connectors between components with mediation facilities for handling heterogeneities

WSMO D2, version 1.0, 20 September 2004



Non-Functional Properties

- Every WSMO elements is described by properties that contain relevant, non-functional aspects of the item
- used for management and element overall description
- **Core Properties:**
  - **Dublin Core Metadata Element Set** plus **version** (evolution support)
  - W3C-recommendations for description type
- **Web Service Specific Properties:**
  - quality aspects and other non-functional information of Web Services
  - used for Service Selection



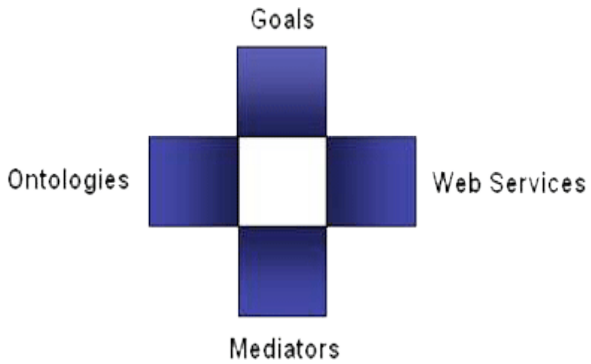
Non-Functional Properties

```
ontology _"http://www.example.org/ontologies/example"
  nfp
    dc#title hasValue "WSML example ontology"
    dc#subject hasValue "family"
    dc#description hasValue "fragments of a family ontology to provide WSML examples"
    dc#contributor hasValue { _"http://homepage.uibk.ac.at/~c703240/foaf.rdf",
      _"http://homepage.uibk.ac.at/~csaa5569/",
      _"http://homepage.uibk.ac.at/~c703239/foaf.rdf",
      _"http://homepage.uibk.ac.at/homepage/~c703319/foaf.rdf" }
    dc#date hasValue _date("2004-11-22")
    dc#format hasValue "text/plain"
    dc#language hasValue "en-US"
    dc#rights hasValue _"http://www.deri.org/privacy.html"
    wsml#version hasValue "$Revision: 1.13 $"
  endnfp
```

WSMO Ontologies

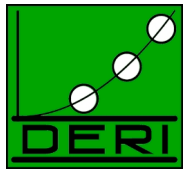
Objectives that a client may have when consulting a Web Service

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



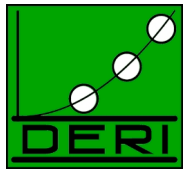
Semantic description of Web Services

Connectors between components with mediation facilities for handling heterogeneities



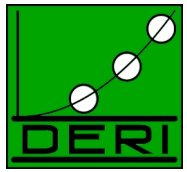
Ontology Specification

- **Non functional properties** (see before)
- **Imported Ontologies** importing existing ontologies where no heterogeneities arise
- **Used mediators:** OO Mediators (ontology import with terminology mismatch handling)
- **'Standard' Ontology Notions:**
  - **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)



Ontology: Example 1/2

```
concept Human
      nonFunctionalProperties
            dc#description hasValue "concept of a human being"
      endNonFunctionalProperties
      hasName ofType foaf#name
      hasParent inverseOf(hasChild) impliesType Human
      hasChild impliesType Human
      hasAncestor transitive impliesType Human
      hasWeight ofType (1) _decimal
      hasWeightInKG ofType (1) _decimal
      hasBirthdate ofType (1) _date
      hasObit ofType (0 1) _date
      hasBirthplace ofType (1) loc#location
      isMarriedTo symmetric impliesType (0 1) Human
      hasCitizenship ofType oo#country
      isAlive ofType (1) _boolean
                  nfp
                        dc#relation hasValue {IsAlive}
                  endnfp
```



Ontology: Example 2/2

```
axiom IsAlive
      definedBy
            ?x[isAlive hasValue _boolean("true")] :-
                  naf ?x[hasObit hasValue ?obit] memberOf Human.
            ?x[isAlive hasValue _boolean("false")]
            impliedBy
                  ?x[hasObit hasValue ?obit] memberOf Human.

 axiom FunctionalDependencyAlive
       definedBy
             !- IsAlive(?x,?y1) and
                        IsAlive(?x,?y2) and ?y1 != ?y2.

concept Man subConceptOf Human
      nfp
            dc#relation hasValue ManDisjointWoman
      endnfp

concept Woman subConceptOf Human
      nfp
            dc#relation hasValue ManDisjointWoman
      endnfp

axiom ManDisjointWoman
      definedBy
            !- ?x memberOf Man and ?x memberOf Woman.
```

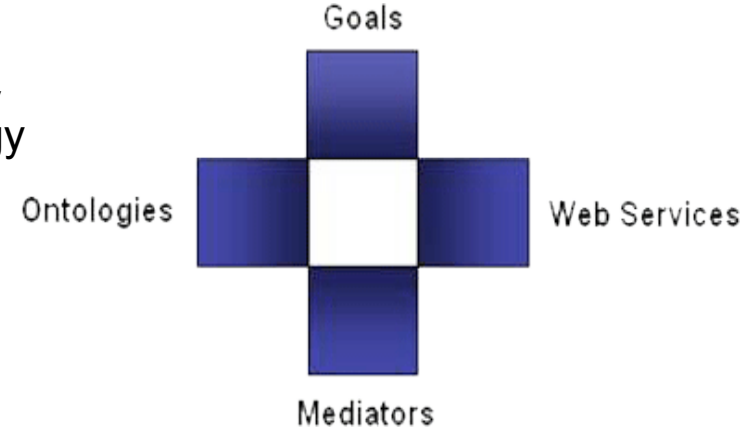
WSMO Capabilities/Interfaces

Requested/provided:
- **Capability** *(functional)*
- **Interfaces** *(usage)*

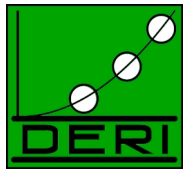
Objectives that a client may have when consulting a Web Service

Semantic description of Web Services:

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

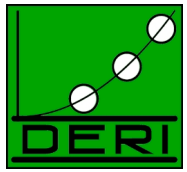


Connectors between components with mediation facilities for handling heterogeneities



Capability Specification:

- **Non functional properties**
- **Imported Ontologies**
- **Used mediators**
  - *OO Mediator:* importing ontologies as terminology definition
  - *WG Mediator:* link to a Goal that is solved by the Web Service
- **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 and work correctly, but not necessarily checked/checkable.
- **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)



Capability - Example

eGovernment: Effect– Service makes a child a German citizen …)

```
capability
      sharedVariables ?child
      precondition
            nonFunctionalProperties
                  dc#description hasValue "The input has to be boy or a girl
                        with birthdate in the past and be born in Germany."
            endNonFunctionalProperties
            definedBy
                  ?child memberOf Child
                        and ?child[hasBirthdate hasValue ?brithdate]
                        and wsml#dateLessThan(?birthdate,wsml#currentDate())
                        and ?child[hasBirthplace hasValue ?location]
                        and ?location[locatedIn hasValue oo#de]
                        or (?child[hasParent hasValue ?parent] and
                              ?parent[hasCitizenship hasValue oo#de] ) .

      assumption
            nonFunctionalProperties
                  dc#description hasValue "The child is not dead"
            endNonFunctionalProperties
            definedBy
                  ?child memberOf Child
                        and naf ?child[hasObit hasValue ?x].

      effect
            nonFunctionalProperties
                  dc#description hasValue "After the registration the child
                        is a German citizen"
            endNonFunctionalProperties
            definedBy
                  ?child memberOf Child
                        and ?child[hasCitizenship hasValue oo#de].
```

WSMO Web Service - Interfaces

- complete item description
- quality aspects
- Web Service Management

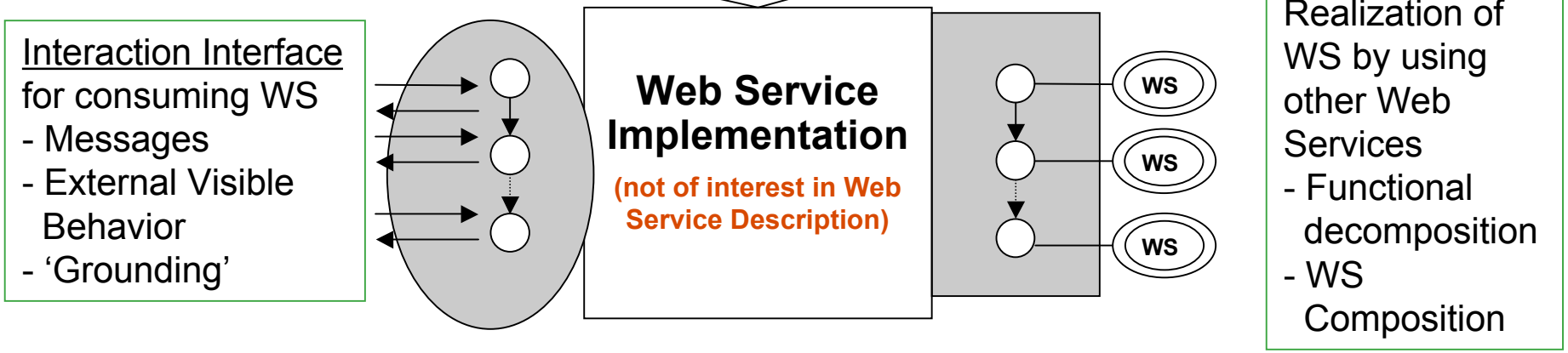
Non-functional Properties

- Advertising of Web Service
- Support for WS Discovery

Capability

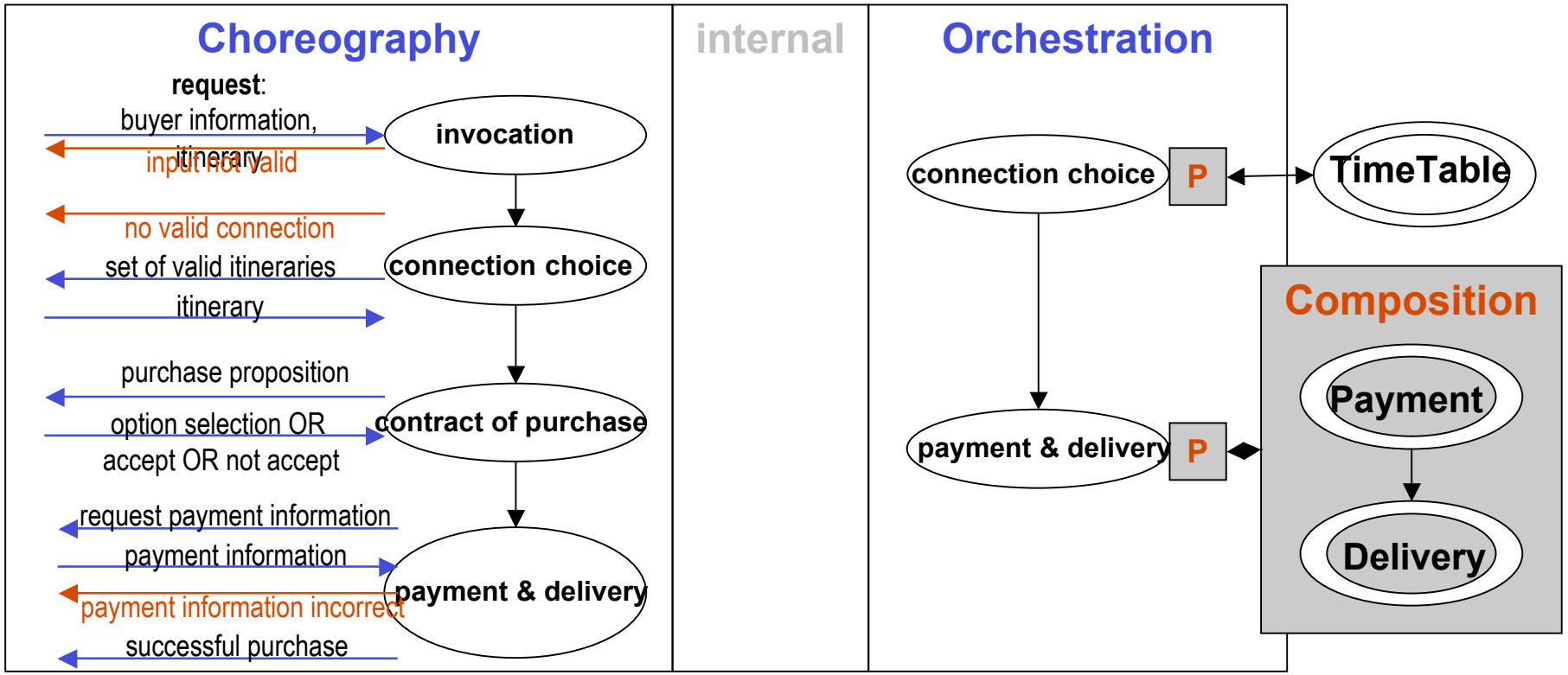
Core + WS-specific

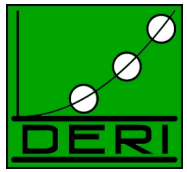
functional description



Choreography --- Interfaces --- **Orchestration**

Web Service Interfaces

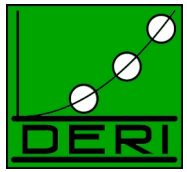




Choreography in WSMO

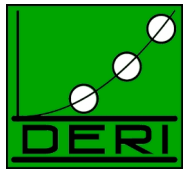
"Interface of Web Service for client-service interaction when consuming the Web Service"

- **External Visible Behavior**
  - those aspects of the workflow of a Web Service where User Interaction is required
  - described by process / workflow constructs
- **Communication Structure**
  - messages sent and received
  - their order (messages are related to activities)



Choreography in WSMO (2)

- **Grounding**
  - concrete communication technology for interaction
  - choreography related errors (e.g. input wrong, message timeout, etc.)
- **Formal Model**
  - allow operations / mediation on Choreographies
  - Formal Basis: Abstract State Machines (ASM)
- Very generic description of a transition system over evolving ontologies:



WSMO Orchestration

"Achieve Web Service Functionality by aggregation of other Web Services"

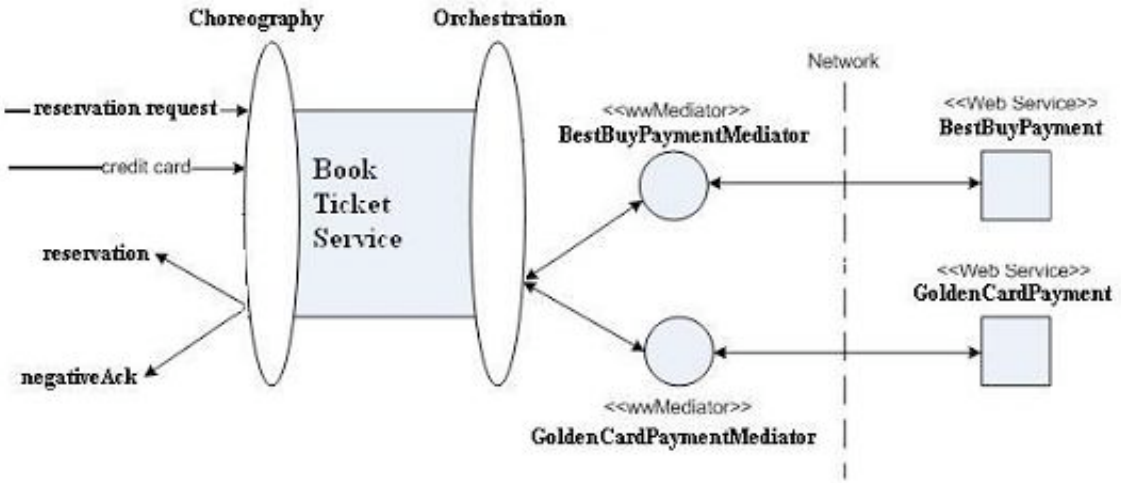
Decomposition of the Web Service functionality into sub functionalities

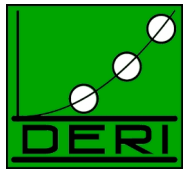
Proxies: Goals as placeholders for used Web Services

- **Orchestration Language**
  - decomposition of Web Service functionality
  - control structure for aggregation of Web Services
- **Web Service Composition**
  - Combine Web Services into higher-level functionality
  - Resolve mismatches occurring between composed Web Services
- **Proxy Technology**
  - Placeholders for used Web Services or goals, linked via Mediators.
  - Facility for applying the Choreography of used Web Services, service templates for composed services

Choreography & orchestration

- Example:





Choregraphy & Orchestration:

```
choreography BookTicketChoreography
  state _"http://example.org/BookTicketInterfaceOntology"
  guardedTransitions BookTicketChoreographyTransitionRules

   if  (reservationRequestInstance [
            reservationItem hasValue ?trip,
            reservationHolder hasValue ?reservationHolder
        ] memberOf bti#reservationRequest
       and
       ?trip memberOf tr#tripFromAustria
       and
       ticketInstance[
          trip hasValue ?trip,
          recordLocatorNumber hasValue ?rln
       ] memberOf tr#ticket
       then
        temporaryReservationInstance[
            reservationItem hasValue ticketInstance,
            reservationHolder hasValue ?reservationHolder
        ] memberOf bti#temporaryReservation

   if (temporaryReservationInstance[
          reservationItem hasValue ticketInstance,
          reservationHolder hasValue ?reservationHolder
       ] memberOf bti#temporaryReservation
       and
       creditCardInstance memberOf bti#creditCard
       and
       po#validCreditCard(creditCardInstance))
   then
        reservationInstance[
          reservationItem hasValue ticketInstance,
          reservationHolder hasValue ?reservationHolder
        ]memberOf bti#reservation

   if (temporaryReservationInstance [
          reservationItem hasValue ticketInstance,
          reservationHolder hasValue ?reservationHolder
       ] memberOf bti#temporaryReservation
       and
       creditCardInstance memberOf bti#creditCard
       and
       neg (po#validCreditCard(creditCardInstance)))
   then
        negativeAcknowledgementInstance memberOf bti#negativeAcknowledgement
```

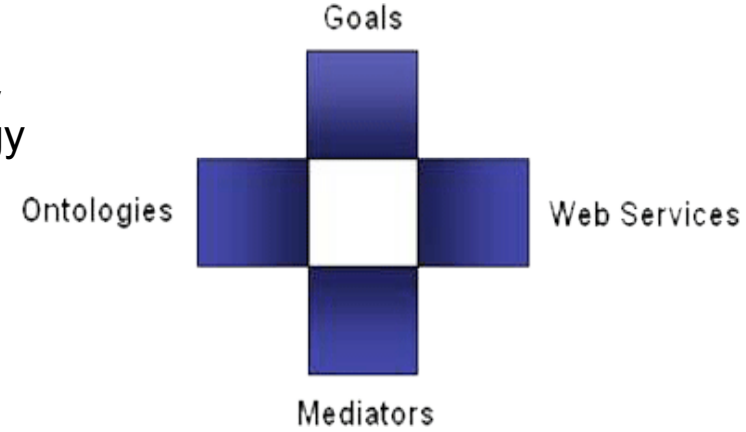
Choregraphy & Orchestration:

```
orchestration BookTicketOrchestration
  state   "http://example.org/BookTicketInterfaceOntology"


 guardedTransitions BookTicketOrchestrationTransitionRules

  if (creditCardInstance[
        type hasValue "BestBuy"
     ] memberOf bti#creditCard
     and
     po#validCreditCard(creditCardInstance))
  then
      _"http://example.org/BestBuyPaymentMediator"

  if (creditCardInstance[
        type hasValue "GoldenCard"
     ] memberOf bti#creditCard
     and
     po#validCreditCard(creditCardInstance))
  then
      _"http://example.org/GoldenCardPaymentMediator"
```

WSMO Goals

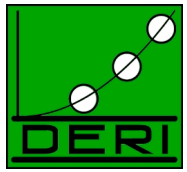
Objectives that a client may have when consulting a Web Service

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

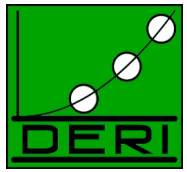


Goals

- **De-coupling of Request and Service**

  Goal-driven Approach, derived from AI rational agent approach
  - Requester formulates objective independent / without regard to services for resolution
  - 'Intelligent' mechanisms detect suitable services for solving the Goal
  - Allows re-use of Goals

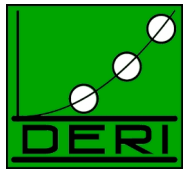
- **Usage of Goals within Semantic Web Services**
  - A Requester, that is an agent (human or machine), defines a Goal to be resolved
  - Web Service Discovery detects suitable Web Services for solving the Goal automatically
  - Goal Resolution Management is realized in implementations



Goal Specification

Goals:

- **have NonFunctionalProperties**
- **import Ontologies**
- **use Mediators**
- **request a Capability**
- **request an Interface**

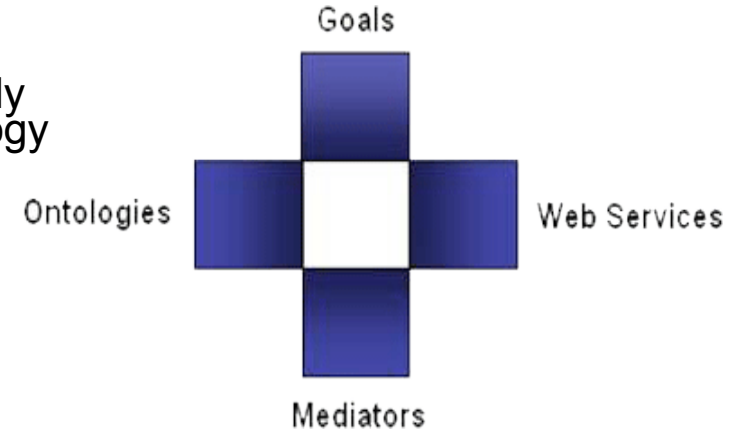


WSMO Standard

WSMO Web Services

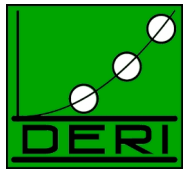
Objectives that a client may have when consulting a Web Service

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



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

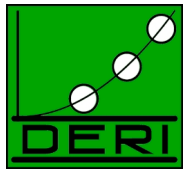
Connectors between components with mediation facilities for handling heterogeneities



Web Service specific Properties

- non-functional information of Web Services:

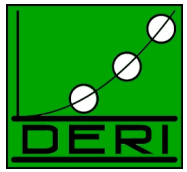
Accuracy
Availability
Financial
Network-related QoS
Performance
Reliability
Robustness
Scalability
Security
Transactional
Trust



Service Specification:

Services :

- **have NonFunctionalProperties**
- **import Ontologies**
- **use Mediators**
- **provides a Capability**
- **provides an Interface**



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

National University of Ireland, Galway
Ollscoil na hÉireann, Gaillimh

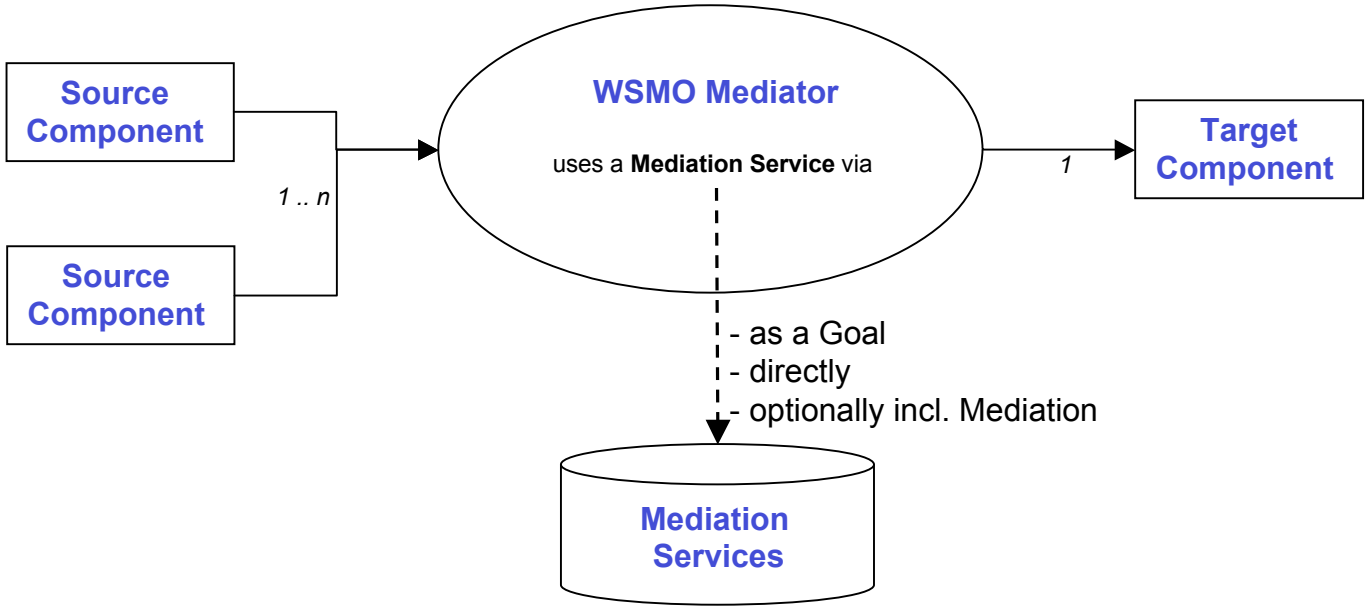
Ongoing work on mediation:

Development of a rule based mapping language for Data Mediation (so-called ooMediators in WSMO).

Protocol Mediation still open: Interesting approaches for composition of WS Interfaces (KnowledgeWeb, Trento!)

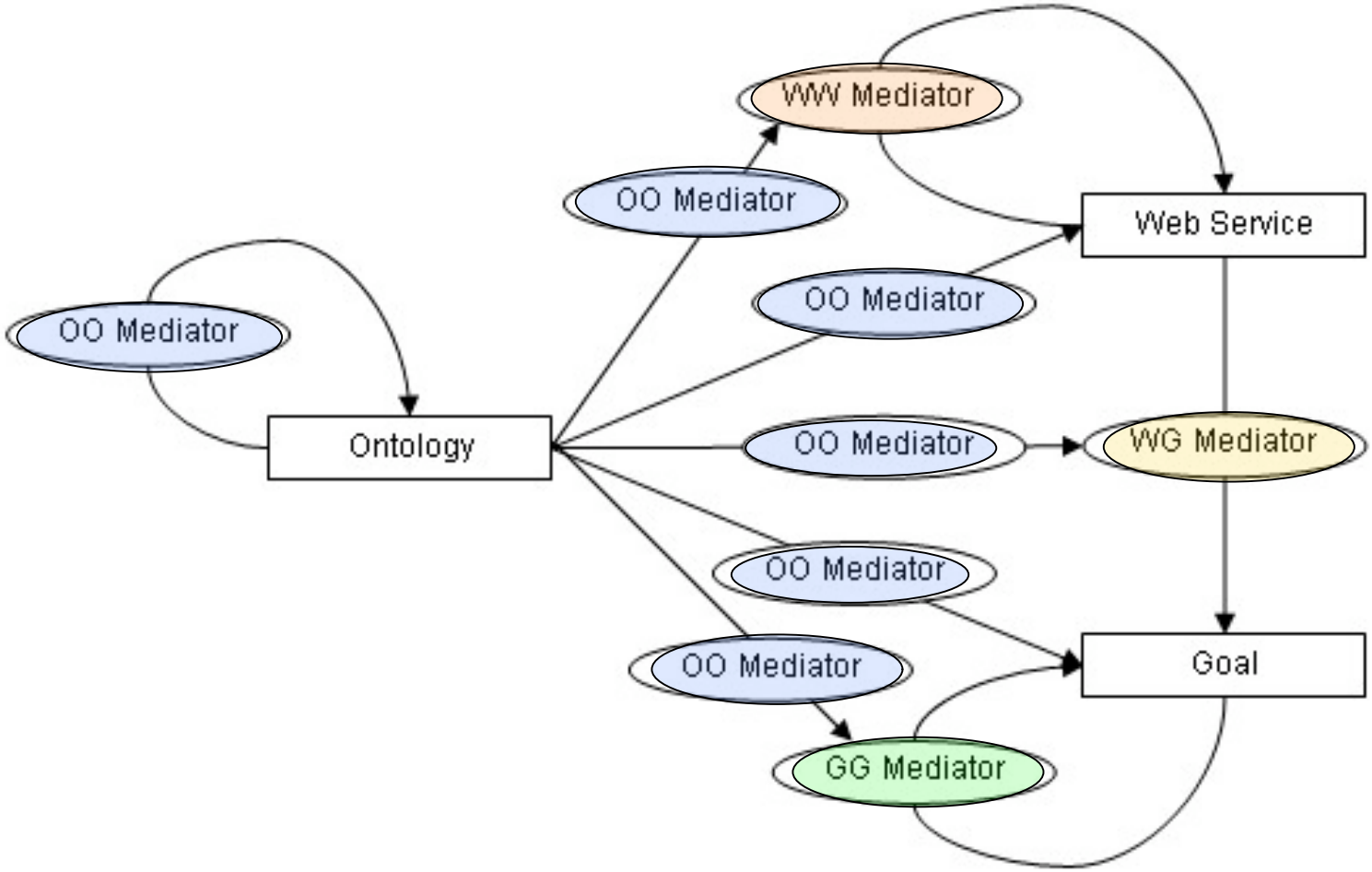
Mediators

- For handling heterogeneity



- Mediator Types: OO, GG, WG, WW

Mediator Usage



Example ooMediator:

OOMediator for importing the OWL Person Ontology into the Trip Reservation Ontology

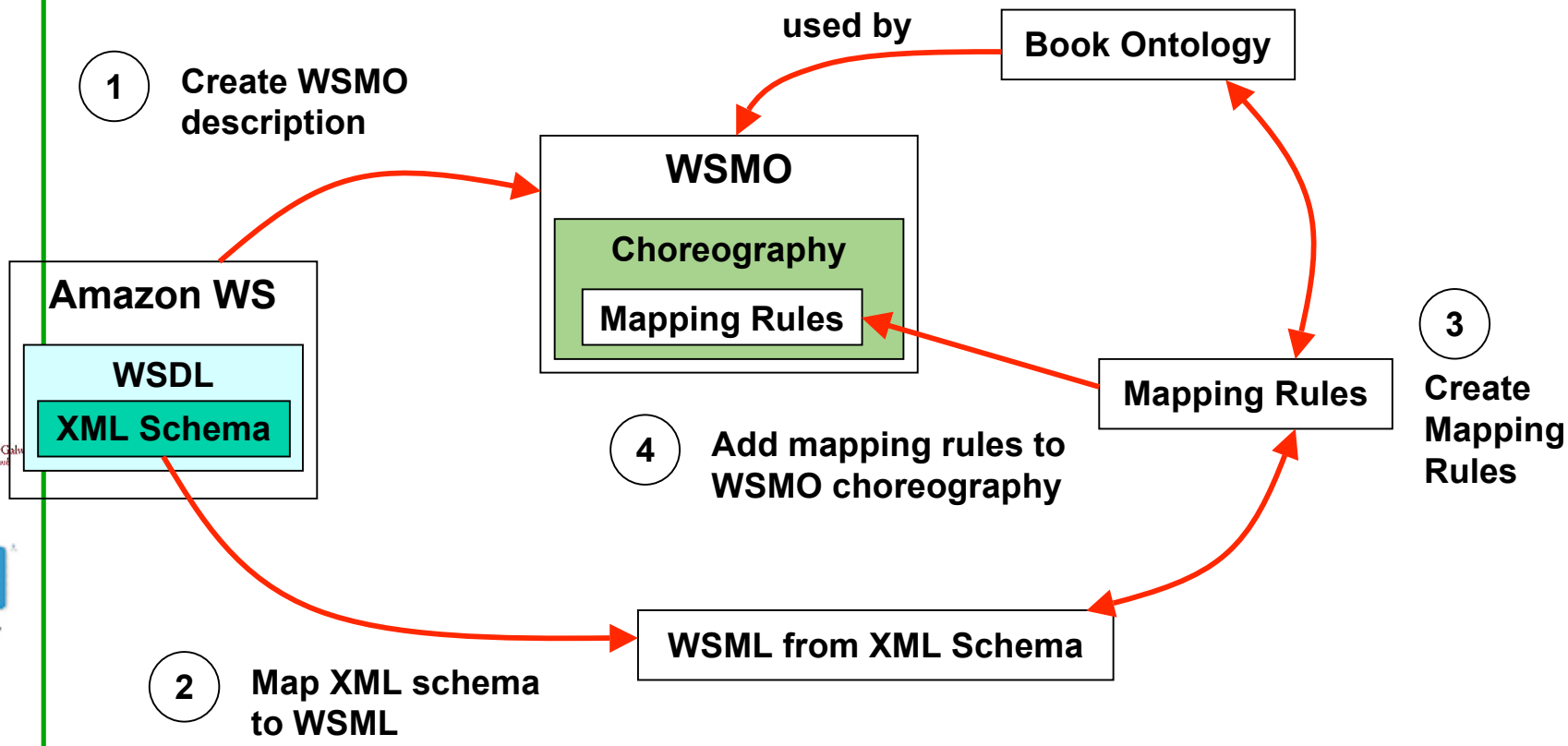
```
namespace{_"http://example.org/mediators#",
  dc    _"http://purl.org/dc/elements/1.1" ,
  wsml _"http://www.wsmo.org/wsml-syntax#"
}
ooMediator _"http://example.org/owlPersonMediator.wsml"
  nonFunctionalProperties
     dc#title hasValue "OO Mediator importing the OWL Person ontology to WSML"
     dc#creator hasValue _"http://example.org/foaf#deri"
     dc#description hasValue "Mediator to import an OWL person ontology into a WSML
          trip reservation ontology"
     dc#publisher hasValue _"http://example.org/foaf#deri"
     dc#contributor hasValue _"http://example.org/foaf#ausen"
     dc#dentifier hasValue _"http://example.org/owlPersonMediator.wsml"
     dc#language hasValue "en-us"
     dc#relation hasValue {_"http://daml.umbc.edu/ontologies/ittalks/person/",
                           _"http://example.org/tripReservationOntology"}
     dc#rights hasValue _"http://www.deri.org/privacy.html"
     wsml#version hasValue "$Revision: 1.14 $"
  endNonFunctionalProperties
  source _"http://daml.umbc.edu/ontologies/ittalks/person/"
  target _"http://example.org/tripReservationOntology"
  usesService  _"http://example.org/OWL2WSML"
```

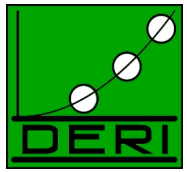
Service Grounding – WSMO

Currently a placeholder in WSMO, mainly investigated by WSMX group (execution environment):

- Deal with existing WSDL services or other grounding technologies:
  - Map from XML Schema used in WSDL to WSML
  - Use existing tools to mediate from WSML to WSML
- Also investigating
  - Using XSLT to map from XML-S of WSDL directly to WSML/XML of ontology used by WSMO description
- Ultimate aim to have Semantic description of interface grounding in the Choreography

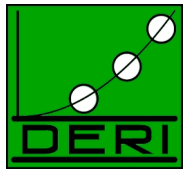
Service Grounding – WSMO





WSMO Perspective

- WSMO provides a **conceptual model** for Web Services and related aspects
  - WSMO separates the different **language specifications layers** (MOF style)
    - Language for defining WSMO is the meta – meta - model in MOF
    - WSMO and WSML are the meta - models in MOF
    - Actual goals, web services, etc. are the model layer in MOF
    - Actual data described by ontologies and exchanged is the information layer in MOF
  - Stress on solving the integration problem
    - Mediation as a key element
  - Languages to cover wide range of scenarios and improve interoperability
  - Relation to industry WS standards
  - All the way from conceptual modelling to usable implementation (WSML, WSMX)
  - Language: WSML: human radable syntax, XML exchange syntax, RDF/XML exchange syntax under consideration



Semantic Representation

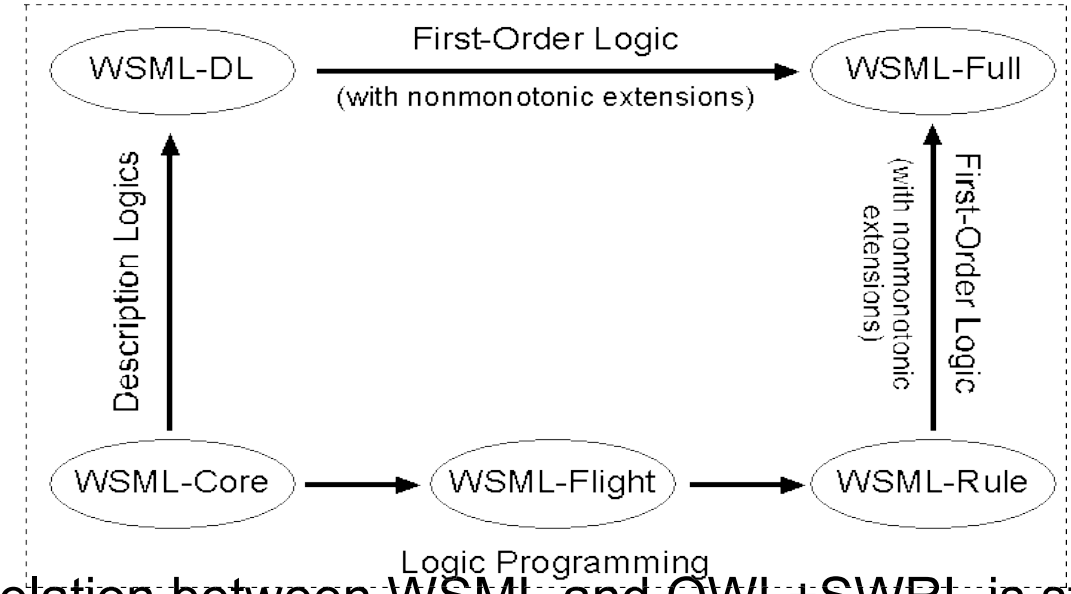
- OWL-S and WSMO adopt a similar view on the need of ontologies and explicit semantics

  but they rely on different logics

  - OWL-S is based on OWL/SWRL
    - OWL represent taxonomical knowledge
    - SWRL provides inference rules
  - 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.

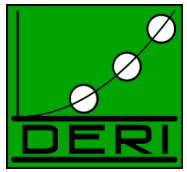
    WSML is a fully-frledged ontology language.

WSML vs OWL



- The relation between WSML and OWL+SWRL is still to be completely worked out:
  - WSML-Core is a subset of OWL Lite (DL ∩ Datalog)
  - WSML-DL is equivalent to OWL DL
  - WSML-Flight (refers to "F-Logic" and "Light" ;-) and extends to the LP variant of F-Logic)

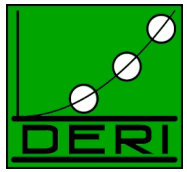
  but for other languages the relation is still unknown.



Relation to Web Services Technology

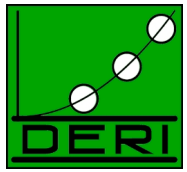
	OWL-S	WSMO	*Web Services Infrastructure*
Discovery ***What it does***	Profile	Web Services (capability)	*UDDI API*
Choreography ***How is done***	Process Model	Orchestration + choreography	*BPEL4WS*
Invocation ***How to invoke***	Grounding+ WSDL/SOAP	Grounding	*WSDL/SOAP*

- OWL-S and WSMO map to UDDI API adding semantic annotation
- OWL-S and WSMO share a default WSDL/SOAP Grounding
- BPEL4WS could be mapped into WSMO orchestration and choreography
- Mapping still unclear at the level of choreography/orchestration
  - In OWL-S, multi-party interaction is obtained through automatic composition and invocation of multiple parties
  - BPEL allows hardcoded representation of many Web services in the same specification.
  - Trade-off: OWL-S support substitution of Web services at run time, such substitution is virtually impossible in BPEL.



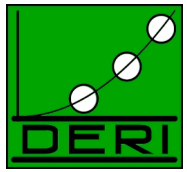
Perspective on Security and Policies

- WSMO distinguishes capabilities, constraints and preferences on both sides [Arroyo et al., 2004]
  - Functional and non-functional
  - Extensions to WSMO required
  - Policies at WSDL level?
  - Must be ensured at execution time
    - Extend WSDL (and others) to include policies and control execution
- Experiments with the representation of policies in WSMO using Peertrust [Lara et al., 2004]
  - Different scope to WS-Policy (trust negotiation)
  - Link to WS-Policy feasible



Conclusion: **How WSMO Addresses WS problems**

- **Discovery**
  - Provide formal representation of capabilities and goal
  - Conceptual model for service discovery
  - Different approaches to web service discovery
- **Composition**
  - Provide formal representation of capabilities and choreographies
- **Invocation**
  - Support any type of WS invocation mechanism
  - Clear separation between WS description and implementation
- **Mediation and Interoperation**
  - Mediators as a key conceptual element
  - Mediation mechanism not dictated
  - (Multiple) formal choreographies + mediation enable interoperation
- **Guaranteeing Security and Policies**
  - No explicit policy and security specification yet
  - Proposed solution will interoperate with WS standards
- The solutions are envisioned maintaining a strong relation with existing WS standards



Related Works:

- METOR-S: extension of WSDL to add ontological concepts to WSDL.
- SWSL: W3C submission under progress, probably overlaps with OWL-S. Semantic Web Service Language… overlap of people ;-)
- Diverse WS Standard proposals, WS-I, WS-Policy, etc.
- WSMO W3C submission also pending!

- W3C workshop on Frameworks for SWS:

  June 9/10, Innsbruck!!!

  http://www.deri.at/events/swsw/index.html

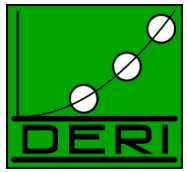
Open Issues:

- Formal semantics of WSMO Interfaces/OWL-S process model
- Formal semantics of the capability of services: OWL-S IOPRs, WSMO Capabilities
- Protocol Mediation
- Grounding… in my opinion not completely solved, neither in WSMO nor OWL-S

- Semantics/Layering and Extensions of Ontology Languages: Local closed world assumption, etc.

- Preferences in Goals
- …

- We are working on it ;-)
- Many challenges!
- Collaboration welcome!
  - WSMO – http://www.wsmo.org
  - WSML - http://www.wsmo.org/wsml
  - WSMX - http://www.wsmx.org
  - Working drafts page - http://www.wsmo.org/



END

Questions? Discussion welcome!