Semantic Web Service Frameworks for Agents

Terry R. Payne
Intelligence, Agents, Multimedia Group
University of Southampton

John Domingue
Knowledge Media Institute
The Open University

Acknowledgements

- There are many who should be thanked for their contribution and assistance in this tutorial, including
  - David de Roure
  - Carole Goble
  - Dieter Fensel
  - Christoph Bussler
  - Sheila McIlraith
  - Jim Hendler
  - Luc Moreau
  - John Domingue
  - Liliana Cabral
  - Enrico Motta
  - Nigel Shadbolt
  - David Martin
  - Sinuhe Arroyo
  - Michael Stollberg
  - Matthew Moran
  - Michal Zaremba
  - Jos de Bruijn
  - Valentina Tamma
- We would also like to thank to all the members of the OWL-S, WSMO, WSML, and WSMX working groups for their advice and input into this tutorial.

Semantic Web Service Frameworks for Agents

Introduction and OWL-S

Terry R. Payne

Intelligence, Agents, Multimedia Group

University of Southampton

Intelligence Agents Multimedia

Agents & Multi-Agent Systems

- Agents & Multi-agent Systems emerged from the field of Distributed AI
  - distribute problem solving across several processes or machines
  - Thus, to coordinate, there is a need to:
    - Communicate
    - Plan
    - Coordinate actions wrt each other
    - …
- Agents emerged as self-contained, autonomous entities that could perform (multiple) services
  - "objects with attitude" *(Jeff Bradshaw)*

Open Agent Systems

- Many Multi-Agent Systems developed by different groups…
  - OAA, Retsina etc
- Lauded for the fact that they could solve various problems using
  - information gathering
  - planning
  - team awareness
  - robustness through redundancy and functional substitutability
- …Provided that all the agents conformed to the same MAS architecture

Combining Multi-Agent Systems

- Early **homogeneous** multi-agent systems (MAS)
  - Could make assumptions on:
    - organization
    - communication protocols
    - ontologies
    - availability of resources
    - benevolence
- Emerging **heterogeneous** MAS
  - Assumptions began to break as different MAS were combined
    - DARPA CoABS Technology Integration Experiments
    - FIPA and Open Agent Systems

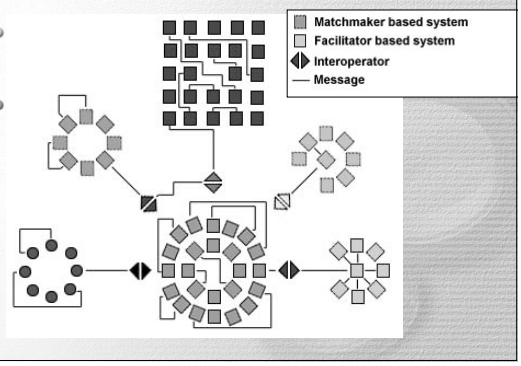
Challenges for Open Agent Systems

- New challenges included:
  - Need to locate agents that provide a given functionality
    - Relaxing the assumption of known models of peers
    - Mechanisms for gathering / managing advertisements
    - Semantic inter-operability required
    - Mechanisms for understanding and coordinating communication
  - Need to set up coalitions of agents to achieve joint goals
    - Relax the assumption of benevolence
    - Planning to determine how goals may be achieved using available resources (e.g. other agents)
    - Negotiation required, to form service level agreements
    - Need to agree on Commitment and Availability
    - Trust, Reputation, Security requirements

… and along came CoABS

- DARPA CoABS program aimed to tack the problem of interoperability between multiple heterogeneous MAS
  - Development of CoABS Grid to support and facilitate interoperability between Agent Communities
  - However, problems still existed in the way knowledge was expressed and shared amongst Agents.
  - Needed to solve the problem of semantic interoperability between agents / services

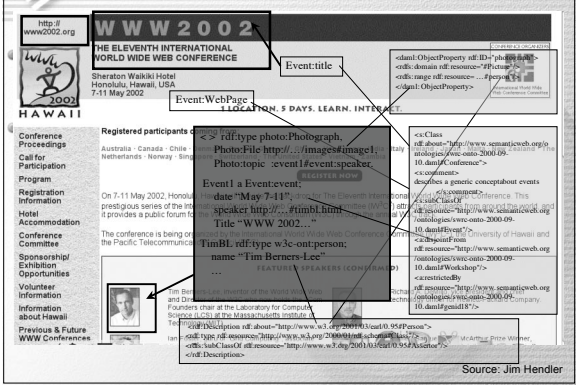
CoABS combining different MAS…



… and along came CoABS

- CoABS DARPA program aimed to tack the problem of interoperability between multiple heterogeneous MAS
  - Development of CoABS Grid to support and facilitate interoperability between Agent Communities
  - However, problems still existed in the way knowledge was expressed and shared amongst Agents.
  - Needed to solve the problem of semantic interoperability between agents / services
- DAML Program
  - Started a couple of years after CoABS
    - Distributed ontologies using Web-based links
    - Based on DLs, but introducing new challenges
    - Helped kick-start the **Semantic Web**

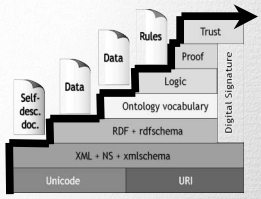
Semantic Web as connective fabric for the Web

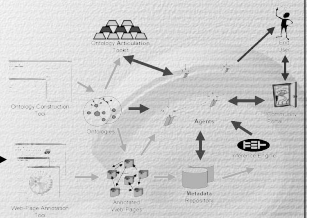


Source: Jim Hendler

Bringing Semantics *to* the Web

- Semantic Web vision





- Semantic Web technologies and tools

Semantic Web

- Not just another Knowledge Base Framework
  - Facts scattered in many locations, referencing each other through URIs
  - Dynamic knowledge source that is typically:
    - Inconsistent
    - Incomplete (open-world)
    - Composed of Several Ontologies
    - Generated by novices and machines, as well as experts
  - Based on a standardized syntax (XML)…
  - … with a superset of standardized expressions (OWL)
  - … supporting various levels of reasoning
    - OWL-Lite / OWL-DL / OWL-Full
    - Others may be defined as necessary
      - *E.g. OWL-Tiny*

Services as a Software Architecture

- Services connect computers and devices with each other using the Internet to **exchange** data and combine data in new ways
- The key to Services is on-the-fly software creation through the use of loosely coupled, reusable software components
- Software can be delivered and paid for as fluid streams of services as opposed to packaged products

New Concept for eWork and eCommerce

New Concept for eWork and eCommerce

"Web Services, are Services accessible via the web."

Numerous white papers

New Concept for eWork and eCommerce

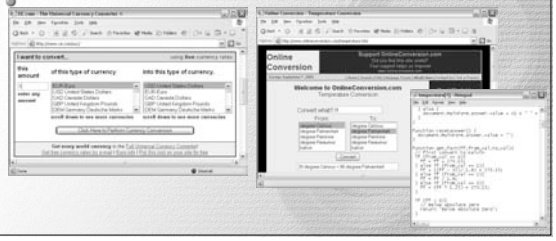
- Business services can be completely decentralized and distributed over the Internet and accessed by a wide variety of communication devices
- The Internet will become a global common platform where organizations and individuals communicate among each other to carry out various commercial activities and to provide value-added services
- The dynamic enterprise and dynamic value chains become achievable and may be even mandatory for competitive advantage

…but…

- …or this would be the case if everybody used exactly the same:
  - Framework
    - For invocation
    - Communication protocol
  - Tools
    - Guaranteeing seamless interoperation
    - Consistent use of middleware
  - Vocabulary
    - Consistent use of language
    - Consistent syntax
- …but this isn't the case!

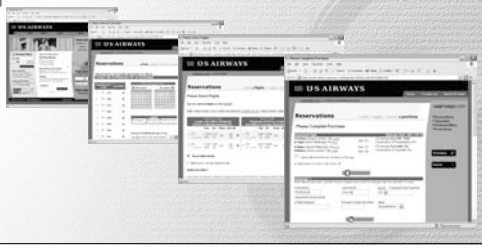
Perspectives on Web Services 1

- Functional (Request-Response) Services
  - Search Engines or conversion services
  - May be implemented simply in Java (e.g. unit conversion) or use GET/POST protocol



Perspectives on Web Services 2

- Conversational Services
  - Online Shopping
  - Require the user to navigate several stages before transaction is complete



Interaction Mechanisms

- Functional – i.e. Request-Response
  - User visits a page and types in request
  - Web service returns an answer
- Conversational – i.e. Solicit-Response interaction
  - *Analogous to being asked questions to refine the request!*
  - Triggered by the user visiting the page, of course
  - Whilst attempting to satisfy user's request, the Web service presents a page containing information and choice
  - User responds with information
  - Ultimately, the service returns the desired response to the initial request

Web Services - Liberating the Machine

- Web Services traditionally have a human interface
  - Required information is presented using forms
  - Humans interpret labels and enter corresponding information
  - Humans interpret resulting information
- Form-based interaction ill-suited for machine comprehension
  - Prior knowledge can be used to prime parsing of pages
    - E.g. screen scraping
  - CGI-based services can ignore presented page and submit a preformed request directly to the server
- Web Services make the implicit specifications explicit!

Human Oriented Services vs Machine Oriented Services

- WWW is organized around URIs, HTML, and HTTP.
  - URIs provide defined ids to refer to elements on the web
    - URLs are machine resolvable, but other URIs may resolve to any object…
  - HTML provides a standardized way to describe document structures
    - allowing browsers to render information for the human reader
  - HTTP defines a protocol to retrieve information from the web.
    - resulting in a near-ubiquitous communication framework
- Not surprisingly, web services require a similar infrastructure around UDDI, WSDL, and SOAP.

Source: Dieter Fensel & Christoph Bussler

Agents vs Web Services

- The WS stack provides a representation specification:
  - **SOAP** messages can support communication
  - **WSDL** - declarative specifications of meaningful messages
    - Typically thought of as describing service interfaces, but a WSDL operation isn't necessarily a service!
  - **BEPL4WS** - describes coordination between operations
    - Still need to determine if the service (with multiple execution paths) can deliver meaningful results given a set of beliefs
  - **WS-Chor** - describes how several services interact to achieve a joint goal
    - Still need to plan across a dynamic space of services to determine a coalition that will achieve this goal
    - This may also require supporting services
      - *(e.g. Agent-based commitment, or GRID reservations)*

One Advantage of building on Web Services

- Web Services have:
  - provided de facto standard for ubiquitous communication
    - http !!!
  - provided a unified base representation language
    - <XML/>
  - provided a universal naming scheme
    - Universal Resources Identifiers
    - Name Spaces
- All this is syntax, but what does it all mean?
  - And if it means something to you, does it mean the same to me?

Agents, Web Services & The Semantic Web

- Semantic Annotations embedded in Web Pages
  - Provide better access to the web
    - Facilitate better navigation within web-pages
    - Support client-side services that utilize the annotations
    - Provide background resources for extended reasoning
  - But may need gathering, monitoring, and managing
    - Automated mechanisms for locating new facts…
    - … and determining the implications
- Web Services benefit from Semantic Annotations
  - Exchanged Data can be richer
  - Descriptions can support enterprise integration
    - Protocol Interoperation
    - Data Representation

Tackling Semantic Interoperability…

- Semantic Interoperability is a major hurdle for
  - Locating Services
    - Different terms used for advertisements and requests
  - Negotiating contracts & communications
    - Different protocols used by different communities when agreeing whether to transact
  - Invoking
    - Constructing valid messages based on the published signature/interface of a service
  - Understanding
    - Interpreting the results of invoking a service
  - Composing Services
    - Constructing plans to achieve meta-goals based on available Services/Agents

Examples of Semantic Mismatch

- **Semantic Mismatch at the Content Level**
  - Provider returns value *Pennsylvania*, but requester only understands two letter state codes (i.e. *PA*)
- **Semantic Mismatch at the Attribute level**
  - Requester needs *rainfall* but provider provides *precipitation*
- **Semantic Mismatch at the level of Units**
  - Requester has value in *inches*, but provider requires *cm*
- **Semantic Mismatch at the Message level**
  - Requester has *length* & *width*, provider requires *area*

Semantic Web Services

Semantic

- Explicitly defined vocabularies (ontologies)
- Expressive representations of concepts and relationships
- Ability to reason and infer new facts, or identify implicit relations

Web

- Distributed entities with (often) well-defined identifiers
- Well defined dissemination/acquisition protocols
- Dynamic, expandable environment

Services

- Reusable objects with one or more capabilities
- Invocable though some communication metaphor

S(WS) vs SW(S)

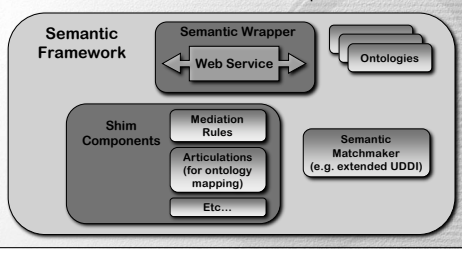
- Semantic + **Web Services**
  - annotating or wrapping **Web Service Technology**
    - Supporting existing WS
      - *Extends existing commercial deployed services*
      - *Builds upon emerging industry-driven standards*
      - *Benefits from commercial tools*
    - Architecture to manage the semantic annotations
      - *Extentions to existing frameworks*
      - *Possibly transparent to the web-service developer*
- **Semantic Web** + Services
  - services supporting, and using the **Semantic Web**
    - Not limited to Web Services
      - *includes Grid, Agent etc*
    - Introduces additional complexity
      - *Different communication or community frameworks*
      - *Different demands or assumptions*

Semantic… Web Services

- Semantic + **Web Services**
  - annotating or wrapping **Web Service Technology**
- Assumes an underlying Web Service Architecture
  - Bounded (constrained) by standards definitions
    - Well defined notions of interface, choreography etc
    - Applications often oriented towards industrially focused domains
      - *Enterprise Applications*
  - Benefits from large user base
  - Realistic near-term potential of technology-transfer
- Two perspectives:
  - Semantically **Annotated** Web Services
  - Semantically **Enabled** Web Services

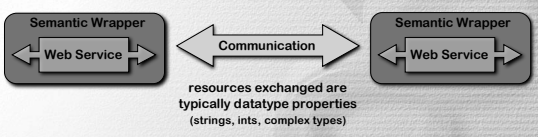
Semantically Annotated … Web Services

- Encapsulates Web Services
  - Can be applied onto existing services without understanding internal logic
  - Developers need no prior knowledge of “Semantics”
  - Semantic Framework / Architecture Required



Semantically Annotated … Web Services

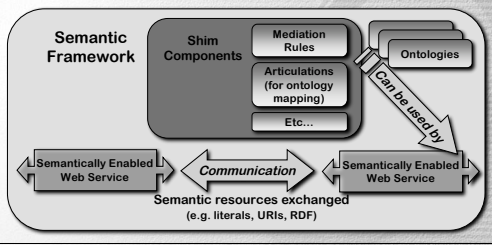
- Encapsulates Web Services
  - Can be applied onto existing services without understanding internal logic
  - Developers need no prior knowledge of “Semantics”
  - Semantic Framework / Architecture Required



resources exchanged are typically datatype properties
(strings, ints, complex types)

Semantically Enabled … Web Services

- Augmented Web Services
  - Services understand Semantic (Web?) resources
  - May have reasoning capabilities
  - Build upon Web Service Technology
    - But existing web services need adapting



Semantic Web … Services

- Services supporting, and using the semantic web
  - Extends any service-oriented, or distributed architecture
    - Grid Services
    - Agent Services
    - Autonomic Services
    - Services within a Pervasive (Ubiquitous) Environment
  - Not bound by the constraints of Web Service technology
    - Can define own protocols and policies
      - *E.g. for dialogues, communication etc*
  - Oriented specifically towards Semantic Web technology
    - Can benefit from "web linkyness"
      - *URIs pointing to other concepts*
      - *Distributed, linked ontologies*
    - May be constrained by reasoning logic
      - *Current descriptions mainly represented in OWL*
      - *Reasoning typically DL based (at the moment)*

Varying Service Characteristics

- Characteristics of a service are dependent on the environment and task they perform
  - Grid Services
    - May be physical or software
    - May be persistent
      - *Need mechanisms for managing multiple contracts*
      - *Need supporting (ancillary) services for functionally related tasks*
    - Invocation duration varies
      - *Typically non-instantaneous*
    - Emphasis on…
      - *Contractural issues, e.g. billing, accountability*
      - *Security issues, authentication, etc*
  - Web Services
  - Agent Services

Varying Service Characteristics

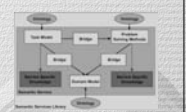
- Characteristics of a service are dependent on the environment and task they perform
  - Grid Services
  - Web Services
    - Typically software based
    - Typically transient instances
      - *Issues of sessions and state are open problems*
    - Well defined structures for defining
      - *Communication & Interfaces*
      - *Orchestration / Choreography*
    - Rapidly evolving standards for more complex requirements
      - *WS-Security, WS-Resources, XACML etc*
    - Invocation duration varies
      - *Typically assumed to be rapid, with acknowlegement being result*
    - Mainly used by commercial sector
  - Agent Services

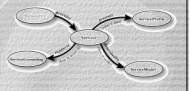
Varying Service Characteristics

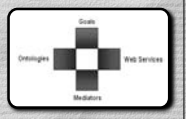
- Characteristics of a service are dependent on the environment and task they perform
  - Grid Services
  - Web Services
  - Agent Services
    - May be physical or software
    - Generally persistent
      - *Agent may present several services / capabilities which may be functionally dependent*
      - *May perform similar tasks concurrently*
    - Not guaranteed to acquiesce when invoked
      - *If request has little utility, agent may choose not to perform task, or may choose to de-commit from an ongoing task*
    - Various
      - *Architectures (both agent and environment)*
      - *Policies, protocols for agent-interaction and dialogues*

Existing & Emerging SWS Frameworks

- UPML
  - Built on the idea of
    - Problem Solving Methods (PSMs) & Tasks
    - Domain Models, Ontologies and Bridges
- OWL-S: OWL for Services
  - Built using OWL-based description logics
  - Views services as having distinct
    - Advertisements / queries (Profile)
    - Workflows (Process Models)
    - Communication / Lower-level protocol binding (Grounding)
- WSMO
  - Complete framework aimed at Enterprise integration
  - Defines its own
    - Language (WSML) - not bound by Description Logics
    - Ontologies (WSMO) for all aspects of semantic web services
    - Architecture and execution framework (WSMX)







Agents, Services and Processes

- Proposed definitions for what constitutes a service / process / agent ...
  - **Agent**:
    - Intelligent Autonomous Entity capable of offering several services (providing different functionalities)
  - **Service**:
    - Encapsulated functionality (or set of functionalities) that achieve a goal (in a variety of ways). Two types…
      - ***Core Service*** *- The main functionality offered*
      - ***Ancillary Services*** *- Additional functionalities that support the core service*
  - **Process**:
    - Declared sub-components within the service
      - *Composite Process can be hierachically decomposed*
      - *Simple Process provides abstraction of a service*
      - *Atomic Process has a defined grounding to support invocation/data passing*

NOTE - Not necessarily the agreed definition across the community!

Main Semantic Web Service Requirements

- Capability descriptions
  - Need to describe what a service does, so that it can be found, compared (to similar ones), and selected, often autonomously
- Workflow descriptions
  - A description of the internal components and the way they are orchestrated is necessary to determine if the service performs the desired task
  - Also used to define the interaction protocol with the exposed funtional components within the workflow
- Interface descriptions
  - Need to define the message structure, syntax, communication endpoints etc
  - Need an invocation framework to support execution of services and communication of data etc

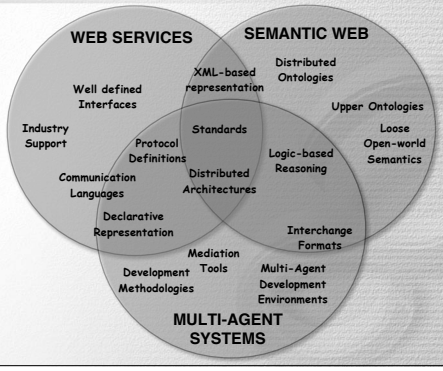
Ancillary Services

- Provide loosely-bound support for a core service
  - Additional services, e.g. reservation, commitment
    - This service should be available at 10am Thursday, when the prior workflow completes. Can I reserve this service?
  - Higher order services, based on parametric call
    - If I invoke this service with these parameters, when will it complete, and what will it cost?
- Such ancillary services necessary for Grid services
- Also exist within many agent frameworks
  - Supporting FIPA behaviors, such as responding to queries
  - Supporting Commitment or Team-based behaviors

Oblex view of Services

- Describing bundles of services
  - Akkermans et al *(service configuration)*
- Supports relationships between services
  - **Core service** - main functionality
  - **Supporting Service** - mandatory additional functionality
  - **Extending Service** - optional additional functionality
- Service bundling common in the business world
  - E.g. Phone or Electricity operators
- Formal concepts applied to agents have received little attention
  - May facilitate planning or service composition?

Agents, Semantic Web & Web Services



Requirements…

- Need a framework to provide semantic descriptions of these services to support
  - Invocation of a previously-unseen service
  - Composition of several services to achieve a goal
  - Planning / Model Checking to determine valid execution paths through the composition
- For Example
  - ***"…generate a workflow to buy all the books within a set, given that only sub-sets of the books are available at any given online book store…"*** **- Jim Hendler**

DAML / OWL Services

- DAML-S (OWL-S) Coalition formed
  - To explore how Distributed (i.e. Web-Based) semantics could be used to describe agents and facilitate knowledge exchange
  - Developed a high level architecture and set of ontologies for services

  - Emphasis moved from **Agents** to **Web Services**…
    - Tighter focus on procedural definitions of the service protocol
    - Emphasis on bounded interface defined explicitly

OWL-S

- **DAML-S**: A **D**ARPA **A**gent **M**arkup **L**anguage for **S**ervices
  - An upper ontology for describing properties & capabilities of agents & (Web) services in an unambiguous, computer interpretable markup language.
- Oriented towards:
  - Discovery
    - Locating services based on a functional description
    - Reasoning about queries using Description Logics
  - Composition
    - Defining the workflow / protocol for execution
    - Support composition (e.g. via planning) and model checking
  - Execution
    - Defined mappings from semantic representation to XML based data definitions
    - Well-defined remote-execution support through Web Services

OWL-S Objectives

- Model services using DAML syntax and description-logics
- Achieve semantic interoperability through tight coupling with Semantic Web standards.
- Automate:
  - WS Discovery
  - WS Invocation
  - WS Selection, Composition & Interoperation
  - WS Execution Monitoring

Automating WS Discovery & Selection

- Support tasks such as
  - Find me an airline that can fly me to Sardinia
- Issues:
  - How are service capabilities defined in the advertisement/request?
    - How are constraints specified?
  - How do mediation/discovery services affect the format of the request specification?

Automating WS Invocation

- Support tasks such as
  - Book flight tickets from Alitalia to arrive on the 9th June.
- Issues:
  - What is the format of the messages sent between service provider and service requester?
  - How are values in incoming messages validated to ensure successful invocation?

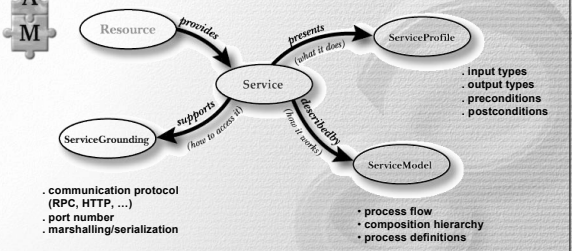
Automating WS Composition & Interoperation

- Support tasks such as:
  - Arrange travel from Pittsburgh to Civitavecchia via Rome.
- Issues:
  - How are composite tasks defined?
  - How are multi-party interactions mediated?
  - How are tasks decomposed
    - Matching with pre-defined composite models?
    - Planning based on known or discovered services?

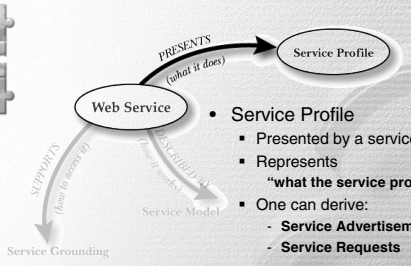
Automating WS Execution Monitoring

- Support tasks such as
  - Have the ferry tickets been reserved yet??
- Issues:
  - How does a service monitor and share its current execution status?
  - How are execution monitoring queries defined?
  - How are failures and aborts handled?

OWL-S Upper Ontology



Presenting Service Profiles



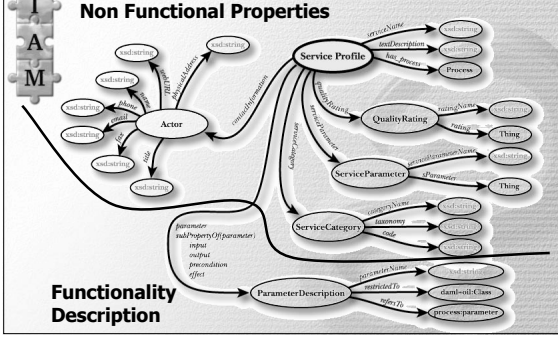
- Service Profile
  - Presented by a service.
  - Represents
    "what the service provides"
  - One can derive:
    - **Service Advertisements**
    - **Service Requests**

Describing the profile

- A profile represents a functional description of the service capabilities
  - Describe:
    - Dataflow properties
      - *Inputs required to invoke the service*
      - *Outputs that are generated by the service*
    - World State properties
      - *Preconditions that should be satisfied*
      - *Effects that will be asserted if the service execution is successful*
- Service metadata is presented
  - Determine additional data that should be used when searching for, or selecting services
    - Quality of Service Parameters
    - References to supporting services
    - etc

DAML-S Service Profile

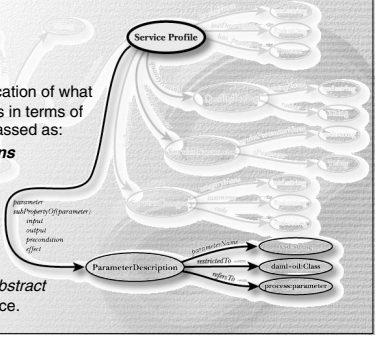
Non Functional Properties



Functionality Description

DAML-S Service Profile

Functionality Description

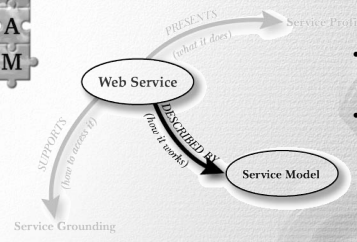


- Functional Specification of what the service provides in terms of ***parameters***, subclassed as:
  - ***preconditions***
  - ***inputs***
  - ***outputs***
  - ***effects***
- Summarizes the *abstract capability* of a service.

DAML-S Service Profile Functionality Description

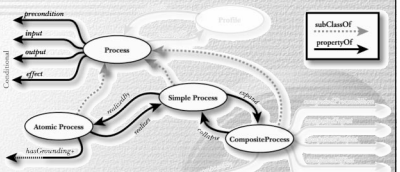
- **Preconditions**
  - Set of conditions that should hold prior to service invocation
- **Inputs**
  - Set of necessary inputs that the requester should provide to invoke the service
- **Outputs**
  - Results that the requester should expect after interaction with the service provider is completed
- **Effects**
  - Set of statements that should hold true if the service is invoked successfully.
  - Often refer to real-world effects
    - Package being delivered, or Credit card being debited

Describing Service Models



- Service Process
  - Describes how a service works.
- Facilitates
  - (automated) Web service invocation
  - composition
  - interoperation
  - monitoring

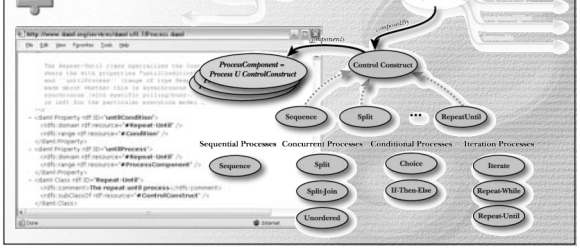
Processes within the Process Model



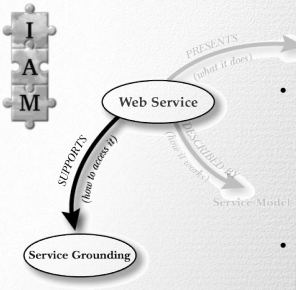
- **Three types of Process**
  - **Atomic**
    - **invokable, bound to grounding**
  - **Simple**
    - **Provides abstraction, encapsulation etc. Not invokable unless realized as an atomic process, or expanded into a composite process**
  - **Composite process**
    - **Models the workflow of several simple or atomic processes**
- **Each process type can express IOPEs**
  - **IOPEs bound to atomic processes, or inferred as computed IOPEs**

Process Model Control Constructs

- **Four types of control construct**
  - **Sequential, concurrent, conditional, and iteration**
- **Test and conditional constructs used to control workflow**



Supporting a Service Grounding

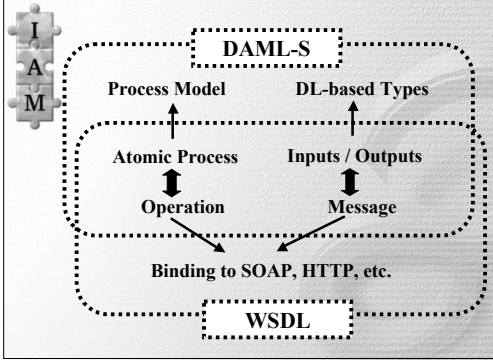


- Service Grounding
  - Provides a specification of service access information.
  - Service Model + Grounding give everything needed for using the service
  - Builds upon **WSDL** to define message structure and physical binding layer
- Specifies:
  - communication protocols, transport mechanisms, agent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

Service Grounding: "How to access it"

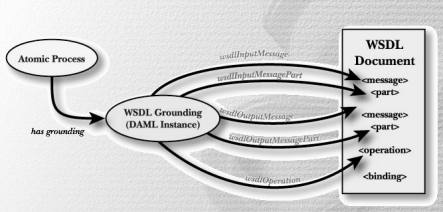
- How to access the service
- Implementation-specific
- Message formatting, transport mechanisms, protocols, serializations of types
- Service Model + Grounding give everything needed for using the service

DAML-S / WSDL Binding



Describing the grounding

- Provide a WSDL file for the service being described
- Identify and map
  - The atomic processes within the process model to the corresponding operations in the WSDL description
  - The inputs & outputs correspond to WSDL messages



An approach to Constructing a service

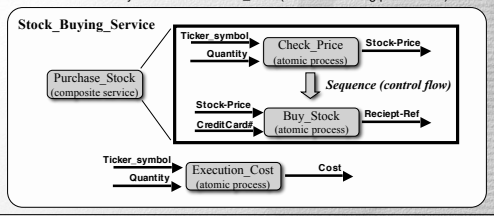
- **Describe the process model**
  - Atomic (functional) or composite (conversational)
  - Determining what to expose
    - Just interaction points
    - Additional process information for reasoning
      - *Transporting Wine trans-continental*
- **Describe the profile and advertise**
  - Expose input & outputs, preconditions and effects
  - Determine relevant metadata
- **Bind to a transport mechanism via the grounding**
  - Provide (or augment existing) WSDL document and bind to it
- **Define the Service concept to link the models together**
  - presents profile
  - describedBy process
  - supports grounding

Ancillary Services

- Provide loosely-bound support for a core service
  - Additional services, e.g. reservation, commitment
    - This service should be available at 10am Thursday, when the prior workflow completes. Can I reserve this service?
  - Higher order services, based on parametric call
    - If I invoke this service with these parameters, when will it complete, and what will it cost?
- Such ancillary services necessary for Grid services
- Also exist within many agent frameworks
  - Supporting FIPA behaviors, such as responding to queries
  - Supporting Commitment or Team-based behaviors

Example - Stock Purchaser

- Stock purchasing service can be used to buy stock. It charges a commission which is a function on the volume of stock bought. Interaction is through a dialog, where the price is provided and credit card details a requested.
  - Service provides:
    - Core Service: Purchase_Stock
    - Ancillary Service: Execution_Cost (based on calling parameters)

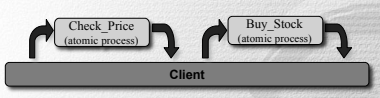


Oblex view of Services

- Describing bundles of services
  - Akkermans et al *(service configuration)*
- Supports relationships between services
  - **Core service** - main functionality
  - **Supporting Service** - mandatory additional functionality
  - **Extending Service** - optional additional functionality
- Service bundling common in the business world
  - E.g. Phone or Electricity operators
- Formal concepts applied to agents have received little attention
  - May facilitate planning or service composition?

Service Dialogue

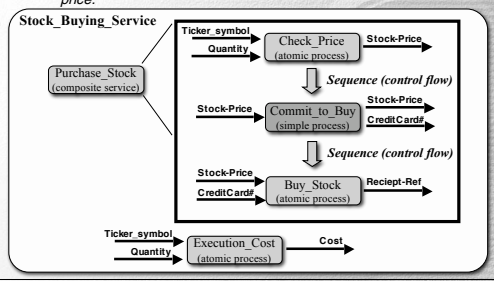
- Individual processes indicate dialogue points when interacting with the core service



- Issue: Client is performing some task based on results of Check_Price - this should be defined
  - Such actions are defined within Agent-based dialog protocols [McBurney & Parsons, 2003]
- Need to use the Simple Process

Example - Modified Stock Purchaser

- Add behavior expected from the client as a simple process
  - *Should be expanded by the client into an atomic service the client provides, that the client uses to decide whether to buy at the given price.*



Planning for dialogues

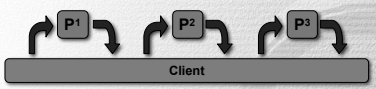
- Planning is necessary to determine execution paths through service dialogue
  - i.e. Model Checking
  - Given a set of beliefs, can the agent:
    1. Understand the different stages of the dialogue
    2. Provide necessary resources to satisfy the dialogue
    3. Trust the service given the resource requirements and client expectations
- Also necessary to:
  - Determine if client can perform the dialogue task (based on its own capabilities)
  - Determine if the task should be delegated to a third party

Planning also used for Service composition

- Several services can be combined to achieve higher level goals
  - e.g. many book stores offer book-buying services. Another may provide a currency conversion service.
  - To achieve Jim's goal, many book-buying services may need to be combined, including those out of the country
    - which need the prices converting to ??? into dollars
- This is the commonly used definition of "Service Composition"
  - May result in several services being choreographed according to the composed model / workflow
  - Different compositions require different control structures…

Service and Process Interaction
Hub-n-spoke

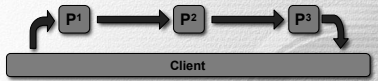
- Conversational Service involves interaction with client throughout the process model
- **Hub-n-Spoke**



- Assumes:
  - the client is always available
    - interaction is pairwise between the client and the service
  - that the client provides mediation between semantic types
  - each service is unaware of the others in the model
    - service is unaware of its role within the composite model
- **This is the approach currently assumed by OWL-S**

Service and Process Interaction
Pipelining

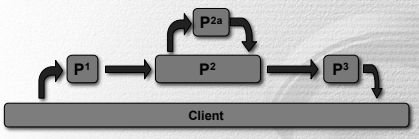
- **Pipelining service** composed of several other services
  - Client provides data at start, and retrieves result at end



- Can support composition of services with "semantic shims"
  - e.g.
    - convert between conceptual types
    - similar to WSMO Ontology mediation
  - May be defined through composition
  - Could be invoked directly by agents
- Don't rely on client being present
- Control may be:
  - Centralised by some enactor
  - Distributed, by passing a partially executed workflow
- **This is the approach often required by GRID Services**

Service and Process Interaction
Delegation

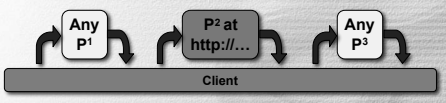
- **Delegation** may occur by an arbitrary service to a third party



- Client may or may not be aware of the delegation
  - Composition may determine 3rd party
    - Important from a security perspective
    - service doing the delegation may be acting on behalf of the client
  - Delegating service may determine 3rd party
    - Client may no longer have control over what parties have access to intermediary results
  - Issues include responsibility, liability
  - This is being explored in Security projects such as the Semantic Firewall

Partially Instantiated Services

- Workflows may be generated by hand, as well as through automated composition
  - Some service instances may be mandatory, and shouldn't be replaced through discovery.
    - E.g. P^2 below
  - Other services can be selected for the plan at plan/execution time
    - E.g. P^1 and P^3 below



Wrap up… Agents, Web Services and OWL-S

- There exists a considerable body of work on Agents
  - encompasses notions of
    - communication
    - negotiation / argumentation / auctions
    - commitment, coalitions, communities
    - evolving behaviour and adaptation
    - autonomous behaviour
  - Different MAS have different ways of defining these
    - both in terms of syntax, and their semantics
- Web Services provide
  - a declarative specification of
    - messages
    - workflows, orchestrations, and choreographies
    - resources, security requirements, contracts
  - Standardized syntax (XML) but lacking:
    - higher level autonomous reasoning, adaptation and coordination
    - formal semantics to support interoperation between services

Wrap up… Agents, Web Services and OWL-S

- OWL-S provides
  - Set of ontologies for defining services, that support:
    - Discovery
    - Workflow analysis, composition
    - Service invocation / interaction
  - Initial design grounded in AI techniques
    - but focus shifted towards augmenting Web Services
- Question: Are requirements of Agents the same as WS?
  - From a simple, functional perspective… perhaps…
  - but complex multi-agent behaviors requires a richer, dynamic representation
    - Many Agent theories have no analogies within the WS community… yet!

Acknowledgements

- I would like to thank many of the people who have contributed and assisted in this tutorial, including
  - David de Roure
  - Carole Goble
  - Dieter Fensel
  - Christoph Bussler
  - Sheila McIlraith
  - Jim Hendler
  - Sinuhe Arroyo
  - Michael Stollberg
  - Matthew Moran
  - Luc Moreau
  - John Domingue
  - Liliana Cabral
  - Enrico Motta
  - Nigel Shadbolt
  - David Martin
  - Michal Zaremba
  - Jos de Bruijn

Web Service Modeling Ontology (WSMO) and IRS-III

John Domingue, Liliana Cabral, Michael Stollberg and Christoph Bussler

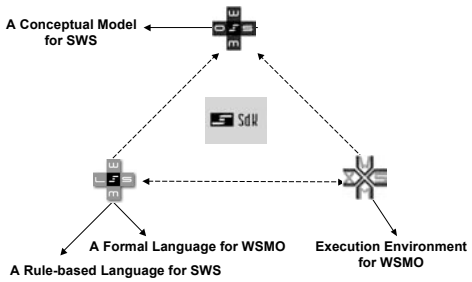
Outline

- WSMO
  - aims & objectives
  - working structure
- WSMO Design Principles
- WSMO Top Level Notions
  - Ontologies
  - Web Services
  - Goals
  - Mediators
- IRS-III Overview
- IRS-III Demo
- Summary

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



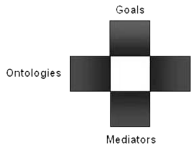
WSMO Design Principles

- **Web Compliance**
- **Ontology-Based**
- **Strict Decoupling**
- **Centrality of Mediation**
- **Ontological Role Separation**
- **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

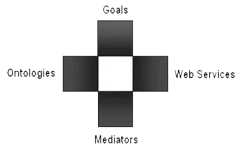
Quality of Service
- Accuracy
- NetworkRelatedQoS
- Performance
- Reliability
- Robustness
- Scalability
- Security
- Transactional
- Trust

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)

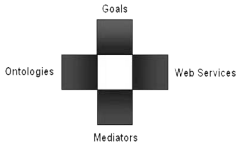
Ontology Elements:

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

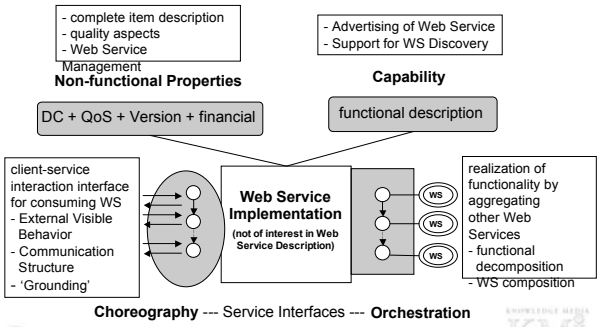
Provide the formally specified terminology of the information used by all other components



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

Connectors between components with mediation facilities for handling heterogeneities

WSMO Web Service Description

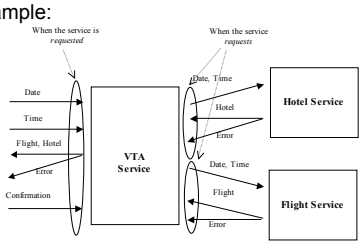


Capability Specification

- **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:



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

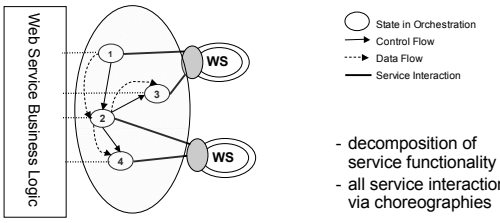
Choreography Aspects

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**
  - concrete 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

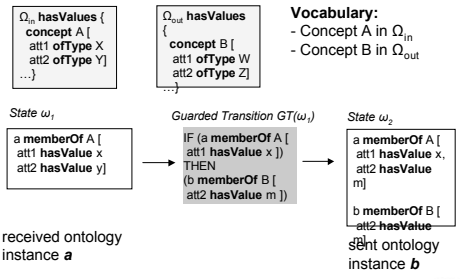
- **Ontologies as data model:**
  - all data elements interchanged are ontology instances
  - service interface = evolving ontology
- **Abstract State Machines (ASM) as formal framework:**
  - dynamics representation: high expressiveness & low ontological commitment
  - core principles: state-based, 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

Service Interface Description Model

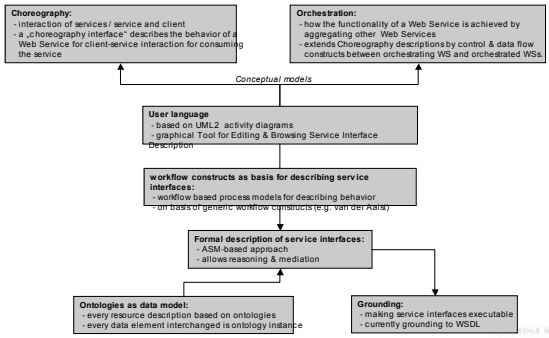
- Vocabulary Ω:
  - ontology schema(s) used in service interface description
  - usage for information interchange: in, out, shared, controlled
- States ω(Ω):
  - a stable status in the information space
  - defined by attribute values of ontology instances
- Guarded Transition GT(ω):
  - state transition
  - general structure: ***if*** (condition) ***then*** (action)
  - different for Choreography and Orchestration

Service Interface Example

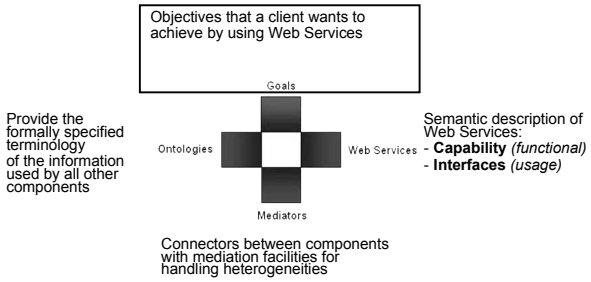
Communication Behavior of a Web Service



Future Directions



WSMO Goals



Goals

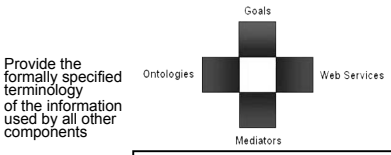
- **Ontological De-coupling of Requester and Provider**
- **Goal-driven Approach**, derived from AI rational agent approach
  - Requester formulates objective independently
  - 'Intelligent' mechanisms detect suitable services for solving the Goal
  - allows re-use of Services for different purposes
- **Usage of Goals within Semantic Web Services**
  - A Requester, 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

- **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

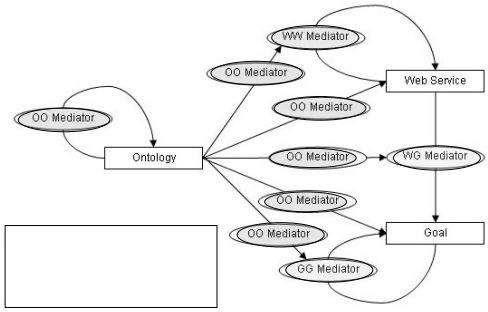
Semantic description of Web Services:
- **Capability** *(functional)*
- **Interfaces** *(usage)*

Connectors between components with mediation facilities for handling heterogeneities

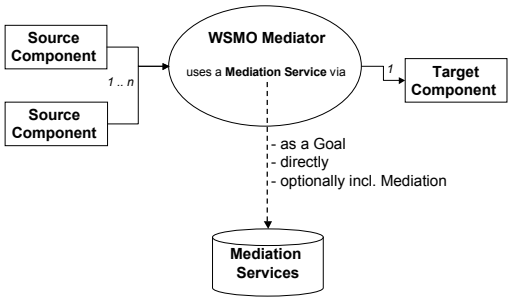
Mediation

- **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

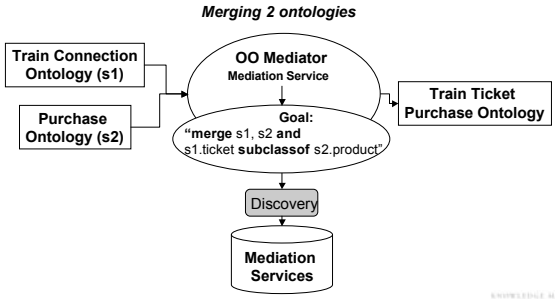
WSMO Mediators Overview



Mediator Structure

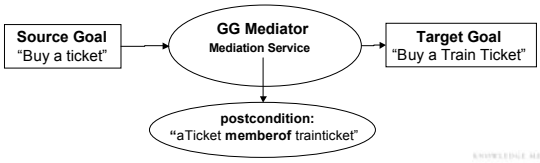


OO Mediator - Example



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**



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

IRS-III:

A framework and platform for building Semantic Web Services

IRSω Internet Reasoning Service

The Internet Reasoning Service is an infrastructure for publishing, locating, executing and composing *Semantic Web Services*

Design Principles

- Ontological separation of User and Web Service Contexts
- Capability Based Invocation
- Ease of Use
- One Click Publishing
- Agnostic to Service Implementation Platform
- Connected to External Environment
- Open
- Complete Descriptions
- Inspectable
- Interoperable with SWS Frameworks and Platforms

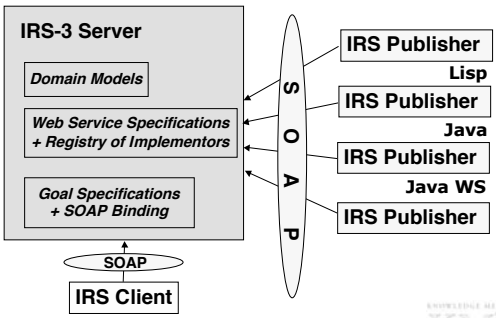
Features of IRS-III (1/2)

- Based on Soap messaging standard
- Provides Java API for client applications
- Provides built-in brokering and service discovery support
- Provides *capability-centred* service invocation

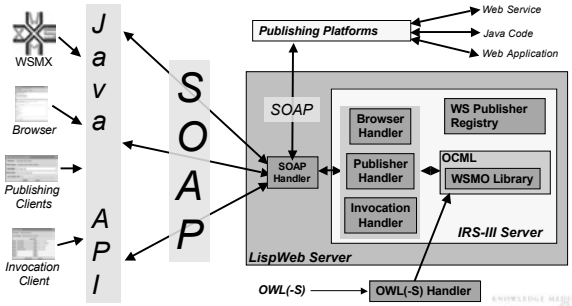
Features of IRS-III (2/2)

- Publishing support for variety of platforms
  - Java, Lisp, Web Applications, Java Web Services
- Enables publication of 'standard code'
  - Provides clever wrappers
  - One-click publishing of web services
- Integrated with standard Web Services world
  - Semantic web service to IRS
  - 'Ordinary' web service

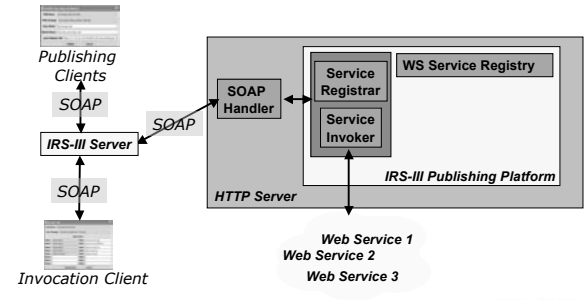
IRS-III Framework



IRS-III Architecture



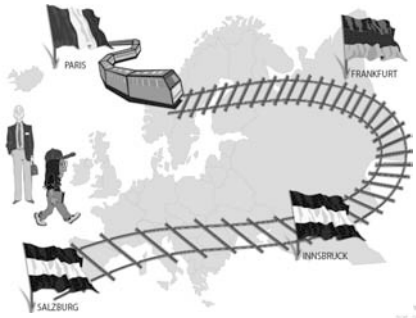
Publishing Platform Architecture



IRS-III/WSMO differences

- Underlying language OCML
- Goals have inputs and outputs
- IRS-III broker finds applicable web services via mediators
  - Used mediator within WS capability
  - Mediator source = goal
- Web services have inputs and outputs 'inherited' from goal descriptions
- Web service selected via assumption (in capability)

European Travel Scenario



IRS Internet Reasoning Service

IRS-III Demo

SWS Creation & Usage Steps

- Create a goal description
  - (e.g. exchange-rate-goal)
  - Add input and output roles
  - Include role type and soap binding
- Create a wg-mediator description
  - Source = goal
  - Possibly add a mediation service
- Create a web service description
  - Used-mediator of WS capability = wg-mediator above
- Specify Operation <-> Lisp function mapping in Choreography Grounding
- Publish against web service description
- Invoke web service by 'achieve goal'

Multiple WS for goal

- Each WS has a mediator for used-mediator slot of capability
  - Some WS may share a mediator
- Define a kappa expression for assumption slot of WS capability
- Kappa expression format
  - (kappa (?goal) <ocml relations>)
- Getting the value of an input role
  - (wsmo-role-value ?goal <role-name>)

Defining a Mediation Service

- Define a wg-mediator
- Source = goal
- Mediation-service = goal for mediation service
- Mediation goal
  - Mediation goal input roles are a subset of goal input roles
- Define mediator and WS as normal

Goal Based Invocation

Invocation

Solve Goal
Goal -> WG Mediator -> WS/Capability/Used-mediator

Instantiate Goal Description	**Web Service Discovery**
Exchange-rate-goal	European-exchange-rate-ws
Has-source-currency: us-dollars	Non-european-exchange-rate-ws
Has-target-currency: pound	European-bank-exchange-rate-ws

WS -> Capability -> Assumption expression	**Mediation**	**Invocation**
Web service selection	**Mediate input values**	**Invoke selected web service**
European-exchange-rate	'$' -> us-dollar	European-exchange-rate

Summary

- WSMO
  - Based on ontologies
  - Goal
  - Web Service
  - Mediator
  - http://www.wsmo.org/
- WSML
- WSMX
- IRS-III
  - Capability based invocation
  - 'One Click' Publishing of web services
  - Open Platform
  - http://kmi.open.ac.uk/projects/irs/

References WSMO

- The central location where WSMO work and papers can be found is WSMO Working Group: http://www.wsmo.org
- WSMO languages – WSML Working Group: http://www.wsml.org
- WSMO implementation
  - WSMX working group : http://www.wsmx.org
  - WSMX open source can be found at: https://sourceforge.net/projects/wsmx/

References WSMO

- [WSMO Specification]: Roman, D.; Lausen, H.; Keller, U. (eds.): ***Web Service Modeling Ontology***, WSMO Working Draft D2, final version 1.2, 13 April 2005.
- [WSMO Primer]: Feier, C. (ed.): ***WSMO Primer***, WSMO Working Draft D3.1, 18 February 2005.
- [WSMO Choreography and Orchestration] Roman, D.; Scicluna, J., Feier, C. (eds.): ***Ontology-based Choreography and Orchestration of WSMO Services***, WSMO Working Draft D14, 01 March 2005.
- [WSMO Use Case] Stollberg, M.; Lausen, H.; Polleres, A.; Lara, R. (ed.): ***WSMO Use Case Modeling and Testing***, WSMO Working Drafts D3.2; D3.3.; D3.4; D3.5, 05 November 2004.
- [WSML] de Bruijn, J. (Ed.): ***The WSML Specification***, WSML Working Draft D16, 03 February 2005.

References WSMO

- [Arroyo et al. 2004] Arroyo, S., Lara, R., Gomez, J. M., Berka, D., Ding, Y. and Fensel, D: "Semantic Aspects of Web Services" in Practical Handbook of Internet Computing. Munindar P. Singh, editor. Chapman Hall and CRC Press, Baton Rouge. 2004.
- [Berners-Lee et al. 2001] Tim Berners-Lee, James Hendler, and Ora Lassila, "The Semantic Web". Scientific American, 284(5):34-43, 2001.
- [Chen et al., 1993] Chen, W., Kifer, M., and Warren, D. S. (1993). HILOG: A foundation for higher-order logic programming. Journal of Logic Programming, 15(3):187-230.
- Domingue, J. Cabral, L., Hakimpour, F., Sell D., and Motta, E., (2004) IRS-III: A Platform and Infrastructure for Creating WSMO-based Semantic Web Services WSMO Implementation Workshop (WIW), Frankfurt, Germany, September,2004
- [Fensel, 2001] Dieter Fensel, "Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce", Springer-Verlag, Berlin, 2001.

References WSMO

- [Gruber, 1993] Thomas R. Gruber, "A Translation Approach to Portable Ontology Specifications", Knowledge Acquisition, 5:199-220, 1993.
- [Grosof et al., 2003] Grosof, B. N., Horrocks, I., Volz, R., and Decker, S. (2003). Description logic programs: Combining logic programs with description logic. In Proc. Intl. Conf. on the World Wide Web (WWW-2003), Budapest, Hungary.
- [Kifer et al., 1995] Kifer, M., Lausen, G., and Wu, J. (1995). Logical foundations of object-oriented and frame-based languages. JACM, 42(4):741-843.
- [Pan and Horrocks, 2004] Pan, J. Z. and Horrocks, I. (2004). OWL-E: Extending OWL with expressive datatype expressions. IMG Technical Report IMG/2004/KR-SW-01/v1.0, Victoria University of Manchester. Available from http://dl-web.man.ac.uk/Doc/IMGTR-OWL-E.pdf.
- [Stencil Group] - www.stencilgroup.com/ideas_scope_200106wsdefined.html

References WSMO

- OWL-- - http://www.wsmo.org/2004/d20/d20.1/
- OWL Flight – http://www.wsmo.org/2004/d20/d20.3/
- [Völz, 2004] Völz, R. (2004). Web Ontology Reasoning with Logic Databases. PhD thesis, AIFB, Karlsruhe.
- WSML-Core – http://www.wsmo.org/2004/d16/d16.7/
- [WSMO Standard] Roman, D.; Lausen, H.; Keller, U. (eds.): Web Service Modeling Ontology - Standard (WSMO - Standard) v 1.0, WSMO Working Draft D2, 16 August 2004.
- [WSMO Choreography] Roman, D.; Stollberg, M.; Vasiliu, L.; Bussler, C.:(eds.): Choreography in WSMO, WSMO Working Draft D14, 17 August 2004.
- [WSMO Orchestration] Roman, D.; Vasiliu, L.; Bussler, C.: (eds.): Orchestration in WSMO, WSMO Working Draft D15, 29 May 2004.
- [WSMO Use Case] Stollberg, M.; Lausen, H.; Polleres, A.; Lara, R. (ed.): WSMO Use Case Modeling and Testing, WSMO Working Draft D3.2, 19 July 2004.

References IRS III tutorial

- J. Domingue, L. Cabral, F. Hakimpour,D. Sell and E. Motta: IRS-III: A Platform and Infrastructure for Creating WSMO-based Semantic Web Services. Proceedings of the Workshop on WSMO Implementations (WIW 2004) Frankfurt, Germany, September 29-30, 2004, CEUR Workshop Proceedings, ISSN 1613-0073, online http://CEUR-WS.org/Vol-113/paper3.pdf.
- J. Domingue and S. Galizia: Towards a Choreography for IRS-III.
- Proceedings of the Workshop on WSMO Implementations (WIW 2004) Frankfurt, Germany, September 29-30, 2004, CEUR Workshop Proceedings, ISSN 1613-0073, online http://CEUR-WS.org/Vol-113/paper7.pdf.
- Cabral, L., Domingue, J., Motta, E., Payne, T. and Hakimpour, F. (2004).
- Approaches to Semantic Web Services: An Overview and Comparisons. In proceedings of the First European Semantic Web Symposium (ESWS2004);
- 10-12 May 2004, Heraklion, Crete, Greece.
- Motta, E., Domingue, J., Cabral, L. and Gaspari, M. (2003) IRS-II: A Framework and Infrastructure for Semantic Web Services. In proceedings of the 2nd International Semantic Web Conference (ISWC2003) 20-23 October 2003, Sundial Resort, Sanibel Island, Florida, USA.

Acknowledgements

We would like to thank to all the members of the **WSMO**, **WSML**, and **WSMX** working groups for their advice and input into this tutorial.

The WSMO work is funded by the European Commission under the projects **DIP**, **Knowledge Web**, **SEKT**, **SWWS**, **AKT** and **Esperonto**; by **Science Foundation Ireland** under the **DERI-Lion** project; and by the Vienna city government under the **CoOperate** program.

The IRS-III work is funded by DIP and AKT