Semantic Web Services Roadmap

DERI Galway and DERI Innsbruck
Jos de Bruin, Chris Bussler, Dieter Fensel, Reto Krummenacher, Holger Lausen, Matthew Moran, and Michal Zaremba
Structure of Presentation

• Introduction – Service-oriented Architectures
• Concepts
  – Semantic Web Services
• Infrastructure
  – Languages
  – Architecture (Service infrastructure)
    • Functionality
    • Design model
    • Components
    • Tools
• Realization
  – Standardization
    • W3C
    • OASIS
  – Use cases/Methods/Best practices
  – Ontologies
• Conclusions
Introduction
Service-oriented Architectures
Introduction: Service-Oriented Architecture (1)

- Currently, computer science is in the next period of abstraction.
- A generation ago we learnt to abstract from hardware and currently we learn to abstract from software in terms of SERVICE oriented architectures (SOA).
- It is the service that counts for a customer and not the specific software or hardware that is used to implement the service.
- In a later stage, we may even talk in terms of problem-oriented architectures (or more positively expressed in terms of problem-solving oriented architectures) because SOAs are biased towards the service provider and not towards the customer that has a problem that needs to be solved.
Introduction: Service-Oriented Architecture (2)

Problem Solving Layer

Computing Layer

Domain Specific Problem Solving Sol’ns
- Call Center Management
- Payroll
- eCommerce
- Etc
- Etc

Applications & Computing infrastructure
- Applications
- SOA
- Ubiquitous Computing
- GRID
- Utility Computing / On Demand

Michael Broodie
Chief Scientist Verizon
Service-oriented architectures will become quickly the leading software paradigm.

However, SOAs will not scale without signification mechanization of
  – service discovery, service adaptation, negotiation, service composition,
    service invocation, and service monitoring; and
  – data and process mediation.

Therefore, machine processable semantics needs to be added to bring SOAs to their full potential.

This is the mission and the vision of WSMOLX!

Development of open standards (languages) and open source architectures and tools that add semantics to service descriptions.
Concepts
Semantic Web Services
Concepts:

- WSMO
- Triple Space
- Semantic Grid
- Standardization
Objectives that a client may have when consulting a Web Service

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

- Connectors between components with mediation facilities for handling heterogeneities

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

Four top level elements – cornerstone of conceptual model

(www.wsmo.org)
WSMO V2.0; topics for model refinement:

- **Goals**
  - Goal repositories, goal decomposition, non-functional properties
- **Semantic Web services**
  - Relationship to WSDL, non-functional properties
- **Mediators**
  - Deeper understanding of nature of OO, GG, WG, WW mediators
- **Ontologies**
  - Develop for various domains (e.g. EDI), measure usage
The big challenge of defining a Semantic Web service

- **Capabilities**
  - What is a service able to do?
  - What are the requirements on the input and output?
  - **Preconditions, Assumptions, Postconditions and Effects need to be defined.**

- **Interfaces**
  - How can a service be accessed?
  - How does a service solve its task?
  - **Choreography and Orchestration of services need to be defined.**
Open issues with respect to the description of Semantic Web services

- What is needed for an orchestration?
- Use cases for the application of orchestrations?
- How to verify conformance of capability, choreography, and orchestration?
Concepts: SWS & Triple Space (1)
Communication platform for Semantic Web services based on Web principles:

“Persistently publish and read semantic data that is denoted by unique identifiers”

Fundamentals:
- Space-based computing – sharing information, knowledge
- RDF triples of the form: <subject,predicate,object>
- URI – Uniform Resource Identifier
Communication platform for Semantic Web services based on Web principles:

“Persistently publish and read semantic data that is denoted by unique identifiers”

Fundamentals:

- Space-based computing – sharing information, knowledge
- RDF triples of the form: <subject,predicate,object>
- URI – Uniform Resource Identifier
Triple Spaces allow for:

- Time autonomy
- Location autonomy
- Reference autonomy
- Data schema autonomy

Triple Spaces provide a communication paradigm for *anonymous, asynchronous* information exchange that ensure the *persistency* and *unique identification* of the communicated semantic data.
Concepts: SWS & Triple Space (5)

Basic entities:
- **Triples**
  - The semantic data exchanged or shared
- **Triple Space**
  - The space used to exchange or share triples
- **Triple Space server**
  - The infrastructure / server hosting Triple Spaces
Concepts: SWS & Grid (1)

Basic Entities:

• Grid
  – Co-ordinated resource sharing over the Web

• Web Services Resource Framework (WSRF)
  – Specifications linking resources to Web services

• Semantic Web Services
  – Provide the Web endpoints for Grid resources
  – Facilitate discovery, composition, mediation

• Non functional properties:
  – Require particular focus e.g. reliability, price, availability
Concepts: SWS & Grid (2)

Conceptual Architecture for Grid:

Grid Interface

Goal Middleware

Goal 1  Goal 2  ...  Goal n

Grid Infrastructure Interface

Grid Infrastructure (SWS Infrastructure + WSRF)

Semantic Grid Services Interface

Capability  Non-func. Properties
Choreography  Grid Service Endpoint
Orchestration  Grid Service

Semantic Grid Services (Semantic Web Services for Grid)

Resources available to the Grid e.g. CPU, Data Storage
Concepts: Standardization

DERI’s approach to tackle the challenges

• **International working groups**
  – WSMO Working Group → define the models
  – WSML Working Group → define the languages
  – WSMX Working Group → define and implement the execution environment

• **Standardization coordination with W3C**
  – Submission of WSMO/WSML/WSMX and WRL to W3C
  – Aim for participation in SWS standardization

• **Standardization coordination with OASIS**
  – Setting up a TC around WSMX

• **Organizer, host and participant in international workshops on Semantic Web services**
  – W3C Workshop on Frameworks for Semantics in Web Services
  – WSMO Implementation Workshop (WIW 2004,5,6 …)
Infrastructure
Languages
A set of concrete languages is needed for the various tasks:

- **Ontology / Rule Languages (static view)**
  - WSML Core
    - efficiency and compatibility
  - WSML DL
    - decidability, open world semantics
  - WSML Rule
    - efficient existing rule engines
  - WSML Full
    - unifying language, theorem proving

- **Languages for dynamics**
  - Transaction Logic over ASMs

- **Mapping languages**
  - for dynamics (process mediation)
  - for data (data mediation)
Infrastructure: Languages – WSML Core

- **WSML Core**
  - Allows basic modeling of Concept, Attributes and Relations
  - Represents the Intersection between Description Logic and Horn Logic
  - Can be used as solid fundament:
    - Allows reuse of existing Deductive Database and Logic Programming research
    - Represents the maximum subset of Description Logic (and OWL) that can be captured with rule languages
  - According to some surveys a high percentage of existing ontologies fall in this segment
**Infrastructure: Languages – WSML DL**

- **WSML DL**
  - Extension of WSML-Core
    - Close to the logical formalism underlying OWL DL
    - Efficient subsumption reasoner are available
    - Entailment is decidable
    - Open world assumption
  - Features not present:
    - Only limited modeling of restrictions (no closed world *constraints*)
    - No arbitrary rules
• WSML Rule
  – Extension of WSML Core
    • Meta modeling (e.g. concepts as instances)
    • Arbitrary rules, function symbols
    • Constraints in the database sense
    • Use of many years of research in the area of logic programming
  – Features not present:
    • Existentials
    • Classical negation (open world)
    • Equality Reasoning
• WSML Full
  – Extension of WSML DL and WSML Rule
    • Based on First Order Logic with non-monotonic extensions
    • Very expressive but entailment is undecidable
      (contains all features of WSML DL and WSML Rule)
  – Semantics of this language is an open research issues
Infrastructure: Languages – Dynamics

• Choreography and orchestration of web services define communication and cooperation of services.
  – Abstract State Machines provide a minimal set of modeling primitives, but are at the same time expressive enough to model any aspect around computation.
  – Transaction Logic is an extension of predicate logic that accounts for state changes based on a declarative semantics.
Mapping and versioning languages

– Develop languages based on the different underlying epistemological models
  • Identify mapping primitives
  • Find characteristics for different meta models
– Identify Patterns for mapping and make them available as libraries or within the language
Infrastructure Architecture
The primary requirements for SWS Architecture:

- **Discovery**
  - Goal-capability matching

- **Negotiation, Adaptation, and Contracting**
  - Goal-capability matching

- **Composition**
  - Composed service execution
  - New service creation on-the-fly

- **Mediation**
  - Resolve data and process mismatches

- **Invocation**
  - Invoke the services according to their choreographies

- **Monitoring**
  - Monitor the execution of a service
Infrastructure: Design Model

SWS ARCHITECTURE
Infrastructure: Design Model
Infrastructure: Reasoners
Infrastructure: Reasoners (1)

• Reasoners for:
  – Description Logic (DL)
  – Rule Language (RL)
  – First-Order Logic (FOL)
  – Abstract State Machine (ASM)
  – Transaction Logic (TL)
Description Logic (DL) Reasoner

• Applied for:
  – WSML-DL reasoning
  – Reasoning in WSML-Core

• Possible use of existing DL-reasoners
  – RACER
  – FaCT++

• Provision of reasoner bindings
• Maybe provision of own reasoner infrastructure
Rule Language (RL) Reasoner

- Applied for:
  - WSML-Rule reasoning
  - Reasoning in WSML-Core

- Possible solutions could be based on:
  - OntoBroker
  - Flora-2

- There exist joint efforts with Ontoprise for the development of an open source inference engine.

- Provision of reasoner bindings
- Provision of own reasoner infrastructure
First-Order Logic (FOL) Reasoner

- Applied for:
  - WSML-Full reasoning

- Possible solutions could be based on:
  - Vampire (University of Manchester)
  - SNARK

- Provision of reasoner bindings
Infrastructure: Reasoners (4)

- Abstract State Machines (ASM)
  - Reasoning about WS compatibility and deadlock freedom
  - Verification and simulation of transition rules given by the choreography of a service description

- Implementation of ASM required
  - Initial ideas from CoreASM (Simon Fraser University, Canada)
    - Control API
    - Scheduler
    - Abstract Storage
    - Interpreter
  - Back-End reasoner
    - WSML/Flora2
Infrastructure: Reasoners (5)

- Transaction Logic (TL) Reasoner based on Flora
Infrastructure: Semantic Repository
Infrastructure: Semantic Repository (1)

• Why?
  – Provide scalable distributed semantic querying
  – Exploit flexibility of semantic data model (difficult with RDBMS)

• What do they offer?
  – Simplification of datastore aggregation
  – Seamless handling of multiple ontologies

• Examples
  – SESAME: http://www.openrdf.org/
Infrastructure: Semantic Repository (2)

- An early trial to use UDDI has not been continued.
- ORDI from Ontotext is currently used in dip for this.
- We are in the process to define a cooperation with the METEOR-S team on this.
Infrastructure: Triple Space
Infrastructure: Triple Space

Triple Space architecture:
• Communication protocol for data exchange
• Communication interfaces
  – e.g., write(triples) and read(query) over HTTP
• Triple Space operations layer
  – Query handling
  – Reasoner bindings
  – Triple Space management (delete, empty, create…)
  – Data Store management (resource bindings…)
• Storage component
  – RDF Store, file system…
Infrastructure: Application Services
Infrastructure: Discovery

Phased approach:

Keyword-based with Natural Language Processing (NLP)

Coarse grained Service and Goal descriptions

Fine grained Service and Goal descriptions

WS

{Keyword}

W1 ... WL

Level of Abstraction

Syntactic

Semantic („Light“)

Semantic („Heavy“)
Infrastructure: Choreography

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

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

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

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

Control Structure for aggregation of other Web Services:

- decomposition of service functionality
- all service interaction via choreographies
Infrastructure: Data Mediation

• Objectives
  – To mediate the interchanged messages part of a communication process
  – To keep the communication process transparent from data representation point of view
  – To have a semi-automatic mediation process

• Assumptions:
  – Ontological approach to Data Mediation
  – Communicating parties express data in terms of an ontology
  – Interchanged messages → ontology instances
  – Ontologies conform to WSMO conceptual model for ontologies
Infrastructure: Process Mediation

- Objectives
  - Mediate between the observable behaviour of communicating services
  - Keep mediation transparent from process representation point of view
  - Some examples requiring process mediation:
Infrastructure: Composition

• Objectives
  – On the fly composition of Web service to satisfy a goal
  – Composition planned based on semantic descriptions
  – Extension of current planning techniques
Objectives

- Service discovery finds provider *claiming* to offer the service
- Negotiate with service over usage
- Goals or services need to be adapted
- Outcome a contract with a Service Level Agreement (SLA)
Infrastructure: Communication

- Objectives
  - Work with established standards to send and receive messages from the Semantic Web Services infrastructure
  - Initially: ground to WSDL
  - Progress to: Web-like communication mechanisms such as Triple Space
Infrastructure: Management

- Objectives
  - Manage the event mechanism of a Service Oriented Architecture
  - Manage the plug-in mechanism for new components
  - Must be lightweight and distributable
Infrastructure: Vertical Services
Infrastructure: Vertical Service

- Services that are required across the architecture
  - Security
    - Authentication and authorisation for data and process
  - Reliability
    - Infrastructure must appear to be always-available
  - Compensation
    - Sometimes may be required to counteract an action already completed by a service in an orchestration
  - Transactionality
    - Well understood in terms of database systems, must be applied to service execution across the SWS infrastructure
Infrastructure: User Tools
Infrastructure: End User Tools

• Management & Monitoring
  – Interface for the addition, update or removal of infrastructure components
  – Interface for the monitoring of the operation of the functional components of the infrastructure
  – Interface for monitoring the execution of individual thread through the infrastructure e.g. goal → service invocation
Infrastructure: Developer Tools
Infrastructure: Management & Monitoring

• Editors
  – Ontology Editor
  – Goal Editor
  – Mapping Editor
  – Process Editor
Realization
Standardization, Use Cases, and Ontologies
Realization: Standardization

• Broad adoption of SWS requires standardization on a global scale
• Initial steps complete and need to be supported further:
  – WSMO Member Submission to World Wide Web Consortium (W3C) accepted June 2005 (WSMO, WSML, WSMX)
    • Expect a leading role in any SWS Working Group
  – Web Rule Language (WRL) is submitted for standardisation in June 2005
    • Expect a leading role in subsequent Rules Working Group
  – Initialize SWS architecture standardisation through the creation of a new OASIS Technical Committee
    • Functional components and execution semantics
• Grid should use standardized SWS as service endpoints
Realization: Use Cases

• Technological basis for realizing the vision of SWS is prerequisite, but more is needed:
  – Illustrate to society what the Semantic Web can provide
  – Give researchers a benchmark to "compare" research results
  – Stimulate current research to a higher final goal

• Like for the Semantic Web Challenge concrete scenarios are needed to benchmark and illustrate technology

• Concrete use cases in industry need to be continued and extended (Integrated Projects with SAP, IBM, etc.)
• Methodologies and Best Practices are needed to ensure adoption
  – can be derived by applying technology to several use cases
  – essential for early adopters to understand the technology and its potential
  – allows exploitation for companies and spin offs
Realization: Ontologies

• **Industrial Strength Ontologies:**
  – EDIFACT, X12
  – eclass
  – UNSPSC

• **Business Process Ontologies:**
  – BPMO (Business Process Modelling Ontology)

• **Goal Description Ontologies:**
  – Methodologies
  – Prototypes
Conclusions
Conclusions

• Service-oriented architectures are an arising software paradigm with big potential for the IT market.
• Bringing service orientation to its full potential requires its combination with semantics to mechanize important aspects and make it scalable.
• This is the vision to which WSMOLX is committed.
• Open standards and open source environments need to be developed.
• You are invited to join this vision since its full-fledged realization requires serious, joined, and world wide efforts.