LINKING SEMANTICS WEB SERVICE EFFORTS

Integrating WSMX and METEOR-S

Kunal Verma[1], Adrian Mocan[2], Michal Zaremba[2],Amit Sheth[1], John A. Miller[1], Christoph Bussler[2]
[1]LSDIS Lab, University of Georgia, Athens, Georgia, USA
[2]DERI, National University of Ireland, Galway

Abstract: The METEOR-S and WSMO projects deal with the lifecycle Semantic Web services and processes. Though, they are separate projects with different approaches, we have identified points of interoperability between them. In this paper, we describe the projects and present the points of interoperability. We intend this work as a starting point for the Semantic Web services community to start building interoperable systems.

Key words:

WSMO, WSMX, METEOR-S, Semantic Web services, WSDL-S, Ontologies, Web service Interoperability, Web service Semantic Annotation

1. INTRODUCTION

With the growth of service oriented architectures, there has been a exponential growth in the number of applications being exposed as Web services. While the industrial focus has so far been on creating open standards for Web services, there is now a greater realization in the industry that the original goals of service oriented architectures, may be not be realized without adequate semantic representation of the entities involved. This is evidenced by creation of committees to formally create models for Service Oriented Architectures [SOA-RM] and Web service choreography [WS-CHOR] and the recent IBM-UGA technical note [WSDL-S].

Researchers in the academic community have been working with declaring semantics of Web services for quite some time – OWL-S [OWL-S], METEOR-S [METEOR-S] and WSMO [WSMO]. In this paper, we will briefly describe two leading projects in this area –METEOR-S and WSMO, and then discuss the points of interoperability between them. We expect this work to act as an starting point for semantic interoperability across different Semantic Web service and Process projects. The rest of the paper is organized as follows. In sections 2 and 3, we provide brief discussions of METEOR-S and WSMO (WSMX) projects. Sections 4 and 5 discuss using WSDL-S as an interoperation point from WSMX and METEOR-S perspectives respectively. Section 6 discusses other points of interoperability and finally section 7 discusses our conclusions and future work.

2. WEB SERVICES EXECUTION ENVIRONMENT – WSMX

Web service Execution Environment (WSMX) is an execution environment which enables discovery, selection, mediation, and invocation of Semantic Web services [Cimpian et al., 2005]. WSMX is based on the conceptual model provided by WSMO, being at the same time a reference implementation of it. It is the scope of WSMX to provide a test bed for WSMO and to prove its viability as a mean to achieve dynamic interoperability of Semantic Web services. In this section, we briefly present the WSMX functionality and its external behavior, followed by a short overview of the WSMX architecture.

WSMX functionalities can be classified in two main categories: first is the functionality that should be part of any environment for Semantic Web services and second, the additional functionality coming from the enterprise system features of the framework. In the first case, the overall WSMX functionality can be seen as an aggregation of the components' functionalities, which are part of the WSMX architecture. In the second case, WSMX offers features such as plugging mechanism that allows the integration of various distributed components, an internal workflow engine capable of executing formal descriptions of the components behavior or a resource manager that enables the persistency of WSMO and non-WSMO data produced during run-time.

The main components that have been already designed and implemented in WSMX, as described in [Cimpian et al., 2005] are: Core Component, Resource Manager, Discovery, Selection, Data and Process Mediator, Communication Manager, Choreography Engine, Web service Modeling Toolkit and Reasoner. The *Core Component* is the central component of the system connecting all the other components and managing the business logic of the system. The *Resource Manger* manages the set of repositories responsible for the persistency of the WSMO and non-WSMO related data flowing through the system. It is offering an uniform and at the same time the only (in the framework) point of access to potentially heterogeneous implementation of such repositories. The *Discovery* component has the role of locating the services that fulfill a specific user request. This task is based on the WSMO conceptual framework for discovery [Keller et al., 2004] which envision three main steps in this process: Goal Discovery, Web service Discovery, and Service Discovery. Currently in WSMX, the Service Discovery covers only the matching of user's goal against service descriptions based on syntactical consideration. In case that more than one suitable services are found, WSMX offers support for choosing only one of them; this operation is performed by the *Selection* component by applying different techniques ranging from simple "always the first" to multi-criteria selection of variants (e.g., Web services non-functional properties as reliability, security, etc.) and interactions with the requester. Two types of mediators are provided by WSMX to resolve the heterogeneity problems on data and process level. *Data mediation* is based on paradigms of ontologies, ontologies mappings and alignment with direct application on instance transformation. The *Process mediation* offers functionality for runtime analysis of two given patterns (i.e., WSMO choreographies) and compensates the possible mismatches that may appear. The *Communication Manager* through its two subcomponents, the Receiver and the Invoker, enables the communication between the requester and the provider of the services. The invocation of services is based on the underlying communication protocol used by the service provider and it is the responsibility of an adapter framework to implement the interactions that require a different communication protocol than SOAP. The *Choreography Engine* has to provide a means to store and retrieve choreography interface definitions, to initiate the communication between the requester and the provider in direct correlation with the results returned by the Process Mediator, and to keep track of the communication state on both the provider and the requester sides. In addition, it has to provide grounding information to the communication manager to enable any ordinary Web service invocation. Even if the *Reasoner* it is not part of the WSMX development effort, a WSML compliant reasoner is required by various components such as Data Mediator, Process Mediator and Discovery. *The Web services Modeling Toolkit* (WSMT) is a framework for rapid creation and deployment of homogenous tools for Semantic Web services. An initial set of tolls includes a WSML Editor for editing WSML and publishing it to WSMO repositories, a WSMX Monitor for monitoring the state of the

WSMX environment, a WSMX Data Mediation tool for creating mappings between ontologies, and a WSMX Management tool for managing the WSMX environment.

WSMX external behavior is described in terms of so-called entry points which represent standard interfaces that enable communication with external entities. There are four mandatory entry points that have to be available in each working instance of the system. Each of these entry points triggers a particular execution semantics which on its turn, selects the set of components to be used for that particular scenario. More details about WSMX Execution Semantics can be found in [Zaremba and Oren, 2005]. As described in [Zaremba et al., 2005] the four possible execution semantics:

- **One-way goal execution**. This entry point allows the realization of a goal without any back and forth interactions. In this simplistic scenario, the requester has to provide a formal description of its goal in WSML and the data required for the invocation and the system will select and execute the service on behalf of the requester. The requester might receive a final confirmation, but this step is optional.
- **Web service Discovery.** A more complex (and realistic) scenario is to consult WSMX about the set of Web services that satisfy a given goal. This entry point implies a synchronous call; the requester provides a goal and WSMX return a set of matching Web services.
- **Send Message.** After the decision on which service to use was already made, a conversation involving back and forth messages between the requester and WSMX has to start. The input parameter is a WSML message that contains a set of ontology instances and references to the Web service to be invoked and to the targeted Choreography (if it is available).
- **Store Entity in the Registry.** This entry point provides an interface for storing the WSMO related entities described in WSML in the repository.

To summarize, WSMX architecture [Zaremba et al., 2005] consists of a set of loosely coupled components. Having various loosely coupled components as part of a software system is one of the fundamental principles of a Service Oriented Architecture (SOA). Self-contained components with well defined functionalities can be easily plugged-in and plugged-out at any time, allowing them to use each others functionalities. Even if WSMX provides a default implementation for all the components in the architecture, following these principles allows third-party component offering the same functionality (or an enhanced functionality) to be easily plugged-in. The WSMX architecture aims to provide descriptions of the external interface and behavior for all the components and for the system as a whole. By this, the system overall functionality is separated from the implementation of particular components. For more details about the WSMX infomodel, the reader can check the WSMX code base at Sourceforge[1]. In the future, WSMX intends to support dynamic execution semantics, which means it will become possible to dynamically load during runtime the intended behavior of the system.

3. METEOR-S

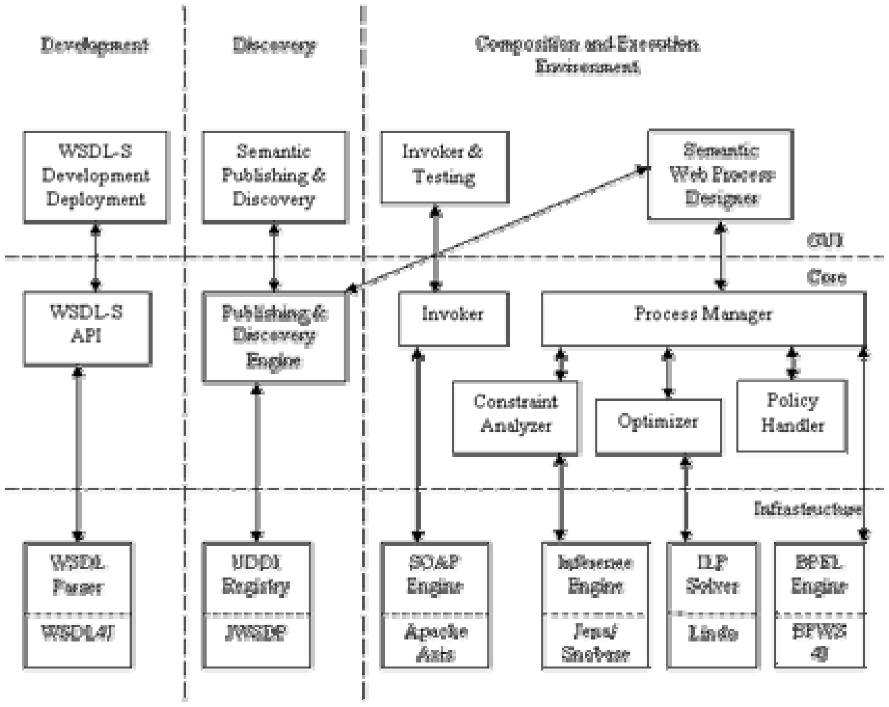

Figure 1. METEOR-S Architecture

The METEOR (Managing End-To-End OpeRations) project at the LSDIS lab addressed the issues related to workflow process management for large-scale, complex workflow process applications in real-world multi-enterprise heterogeneous computing environments [METEOR]. The follow-on project, called METEOR-S endeavors to define and support the complete

[1] WSMX code base at Sourceforge http://sourceforge.net/projects/wsmx

lifecycle of Semantic Web processes [METEOR-S] using four kinds of semantics – data, functional, non-functional and execution semantics. The data semantics describe the data (inputs/outputs) of the Web services. The functional semantics describe the functionality of a Web services (what it does), the non-functional semantics describes the non-functional aspects like quality of service and business rules and the execution semantics models the behavior of Web services and processes.

An architectural overview of METEOR-S is presented in figure 1. The key steps in a life cycle of a Web process are:

- Development of Semantic Web services [Sivashanmugam et al., 2003; Rajasekaran et al., 2004]
- Publication and Discovery of services [Verma et al., 2005]
- Composition of Web processes [Sivashanmugam et al., 2004]
  - Optimization and Constraint analysis [Aggarwal et al., 2004]
  - Runtime Environment for Semantic Web Processes [Verma et al., 2004]

The development module provides a GUI based tool for creating Semantic Web services representing using WSDL-S. The tool provides support for semi-automatic and manual annotation of existing Web services or source code with domain ontologies. WSDL-S is a joint specification with IBM and LSDIS Lab for adding semantic annotation to WSDL. The publication and discovery (MWSDI) module provides support for semantic publication and discovery of Web services. It provides support for discovery in a federation of registries as well as a semantic publication and discovery layer over UDDI. The composition module consists of two main sub-modules – the constraint analysis and optimization sub-module and the execution environment. The constraint analysis and optimization sub-modules deal with correctness and optimization of the process on the basis of quality service constraints. The execution environment provides proxy based dynamic binding support to BPWS4J execution engine for BPEL4WS. A number of METEOR-S tools, including the process designer and Web service developer, are available as open source software at http://lsdis.cs.uga.edu/Projects/METEOR-S/Downloads/.

4. USING WSDL-S FOR INTEROPERABILITY – WSMX PERSPECTIVE

One of the most important aspects in this context is how to link the emerging semantic-related approaches with the already existing standards and technologies. That is, WSMO and METEOR-S propose solutions towards Semantic Web services, but it is important to keep in mind that (most of) the existing Web services have to be invoked using SOAP and their interfaces are described using WSDL. Therefore, grounding is about filling the gap between the WSMO semantic descriptions of Web services and the classical technologies used today. Basically, two main tasks can be distinguished: first to determine which WSDL operation corresponds to the piece of functionality to be consumed (semantically described in the WSMO Web service's capability); second to prepare the data to be used in the invocation as required by the type's definition of inputs in the invoked operation.

WSMX has been confronted with this problem even from the first release when a hard-coded solution was adopted to map semantic descriptions of the Web services with specific WSDL operations. As WSMX became more mature, dynamic and coherent solution were required to enable the full integration of classical Web services in this framework for semantic Web services. The solution envisioned implies the creation of a set of transformations between a given source XML Schema (or even directly between XML data) and a target WSMO ontology. These transformations could take place either on the XML level, by using XSLT [XSLT] for example, or at ontological level by using an ontology mapping language (for example as the one proposed by [de Bruijn et al., 2004]). In the first case, an XML serialization for the ontology mapping language is required (i.e. WSML/XML [WSML, 2005]) while in the second case a "lifting" operation is applied to the XML Schema to create a corresponding WSMO ontology. The first approach, even if it can suffice for some application requirements, seems unnatural in a context where all the efforts are concentrated towards Semantic Web and Semantic Web services. At the same time, the second approach acts on a semantic level, but its effectiveness depends on the techniques used in the lifting and ontology mapping processes which might represent two sensitive points to errors and loss of information. As a consequence, the solution proposed by LSDIS-IBM group as part of the METEOR-S project could represent a good alternative to either of the approaches presented above.

WSDL-S is a lightweight approach for adding semantics to Web services [WSDL-S]. It allows semantic representation of inputs, outputs, preconditions and effects of Web service operations, by adding extension to WSDL, which is a well accepted standard in the industry. In principle, WSDL-S allows semantic annotations using domain models, which are agnostic to the domain representation language. It means that WSMO ontologies can be used in the annotation process and directly link the WSMO semantic description of Semantic Web services with WSDL descriptions. In this way, the annotations of the inputs and outputs in WSDL will represent concepts in a WSMO ontology and the preconditions and effects associated with WSDL operations will point to preconditions and effects from the definition of a WSMO Web service. As Web service specifications in WSMO refer to assumptions and post-conditions as well, additional extensions to the WSDL operations are required to accommodate these features too - the new extensions would be similar with the existing ones for preconditions and effects. By this, one Web service operation will correspond to a Semantic Web service as described in WSMO; if it is desired that more than one operations to be associated to a Semantic Web service than additional annotations are required to link a specific operation with elements described in the choreography of the Web service (for more details about WSMO choreography see [Roman et al., 2005]).

5. USING WSDL-S FOR INTEROPERABILITY – METEOR-S PERSPECTIVE

So far METEOR-S has been limited to the use of OWL ontologies and SWRL rules for modeling all types of semantics. While OWL ontologies are ideal for representing the data semantics of Web services (inputs/outputs), they are limited in representing preconditions and effects, which is critical for representation of the functional semantics (what a Web service does) of a Web service. WSMO represents the functionality of Web services using pre and post conditions specified using F-logic [Keller et al., 2004]. They have recently been working on the WSML [WSML, 2005] family of languages:

- WSML-Core (intersection of Description Logic and Horn Logic)
- WSML-DL (extends WSML-Core to an expressive Description Logic)
- WSML-Flight (extends WSML-Core in the direction of Logic Programming)
- WSML-Rule (extends WSML-Flight to a fully-fledged Logic Programming language)
- WSML-Full (unifies all WSML variants under a common First-Order umbrella)

These languages with their varying computability and expressivity properties provide interesting alternatives to OWL. Concretely, we will investigate creating use cases with WSML annotations in the WSDL-S specification [WSDL-S] and use WSML reasoners for discovery in MWSDI.

6. OTHER POINTS OF INTEROPERABILITY

6.1 Semantic Registry for Publication and Discovery of Web services

The main purpose of the service discovery in WSMX infrastructure is to provide functionality on matching of usable SWS from service providers with the goals of service requesters. When matching SWS, a number of possible services could be discovered whose capabilities satisfy criteria specified in a goal of a requester. Normally, several SWS could be returned from this step. Additionally, WSMX system provides negotiation component allowing refinement the list of services based on their functional properties, and with Selector component enabling to select a service based on its non-functional properties. In the current approach, WSMX publishes services to a Resource Manager together with all the WSMO and non WSMO related entities (non WSMO related entities are the one required by the run environment). Peer-to-peer infrastructure of semantically enhanced registries provided by METEOR-S [Verma at al., 2005] seems to be a promising solution to address WSMX requirements for the distributed persistent storage.

The current infrastructure of WSMX defines a Resource Manager to be responsible for management of repositories to store definitions of any WSMO (Semantic Web services, goals, ontologies and mediators) and non-WSMO related objects (e.g. mappings, messages, WSDL definitions, etc.). Depending on the scope of information stored in the underlying registries, WSMX distinguishes registries to be local or global. Information stored in the local registry is relevant for domain-specific operations (e.g., most of the system run time information will be available to particular components, not to everybody) whereas, the global registry could be shared across several domains (e.g. registries of SWS or Goals descriptions). While both stores data, the accessibility to this data might considerably differ between registries. The Resource Manager remains the only entry point for them (it is not possible to access any of the registries directly, but only through the Resource Manager). In our current approach, the Resource Manager is just a placeholder for the future peer-to-peer infrastructure of global registries and the implementation so far is simply limited to a single relational database. To build the peer-to-peer infrastructure for global registries, we might either assume replication of WSMO entities across all registries and data stores or some sophisticated storage mechanism allowing us to find WSMO entities without replicating them. If we choose the first option, the large growth in number of global registries for WSMO description would make this replication impractical. The second option must allow organizing registries to support semantic publication and discovery of services. Such an infrastructure called METEOR-S Web services Discovery Infrastructure (MWSDI) has been already proposed by METEOR-S and with some modification would suit WSMX requirements.

METEOR-S proposes a specialized ontology called a Registries Ontology at the registries level. This ontology aims to map each registry to a specific domain. By simply providing an obligatory property describing the type of the registry we should easily classify WSMX registries as Web services, Ontologies, Goal and Mediators registries. Additionally to it, we could use the METEOR-S mechanism to provide a richer semantic description to distinguish the types of entities stored in the registry, e.g., we could have Ontology registries with EDI related ontologies, or Goal registries with goals from travel industry.

The Web services Repository in WSMX deals with semantic description of Web services, such as their capabilities (pre-conditions, post-conditions, assumptions and effects), interfaces (choreography and orchestration) and non-functional properties including both general as well as Web service specific. All information stored in the Web services Repository is related to case scenarios in which this information is to be used, such as discovery of services or monitoring of services. To filter Web services before returning them to WSMX discovery component, we could use the Registry Ontology from MWSDI.

The goals repository deals with semantic description of general goals. A WSMO Goal in most cases expresses the “wish” that a user wants to achieve. Some of the goals may be reusable and provided as templates and therefore should be published using the repository. Non reusable or frequently updated goals should be stored at the service requester side or other locations. The goal repository can thus contain predefined goals constructed by domain experts as well as user specific goals.

The ontology repository deals with ontologies to be stored in the registry describing semantics of particular domains. Any components might wish to consult ontology, but in most of the cases ontologies will be used by mediator related components to overcome data and process heterogeneity problems. Ontologies stored by WSMX are expressed in terms of WSML, but we consider also the case that the syntactical adaptation from any ontology syntax is possible, before storing them. WSDI Registry Ontology would be used to classify ontologies before storing them.

The mediator repository deals with mediators to be stored in the registry. WSMO mediator has become a standard proposal covering various approaches for data and process mediation. While mediators in WSMO are still underspecified, we already consider storing them in repository for possible use of the data and process mediation components.

During run-time of the system a lot of system specific data is produced. Also components might generate data, which should be persistent. The data repository allows us to store any system data and any component specific data required for correct system execution. We consider that the Data repository should remain outside WSDI infrastructure.

Another addition to current WSDI infrastructure would be the possibility to extend it towards non UDDI registries. Although UDDI is considered a standard for Web services registries, its application is very limited and the current WSMO entities could not be simply stored there. More practical from WSMO perspective would be ebXML registries allowing persisting any arbitrary objects. We might also want to continue with our current approach to Resource Manager and base our registries on relational databases, while still providing a peer-to-peer infrastructure to access the data.

6.2 Collaboration on Quality of Service Ontology

Quality of service of Web processes and processes has been an area of active research for the METEOR-S project. Our initial work was on QoS aggregation [Cardoso, 2004] based on quantitative models. Our current work focuses on optimization of Web processes using Integer Linear Programming (ILP)[Aggarwal et al., 2004] and semantic representation of quality of service using OWL ontologies. The initial version of the METEOR-S quality of service ontology has been limited to quantitative criteria like time, cost, and reliability. As the METEOR-S and WSMX teams work with their industrial partners (IBM, HP, etc.) on creating real world use cases for Semantic Web services and processes, they can work on jointly building a more comprehensive QoS ontology, which represents not quantitative criteria but also qualitative criteria and business rules.

7. CONCLUSIONS AND FUTURE WORK

A key focus of Semantic Web services projects is on interoperability of different Web services with the help of ontologies. Hence, it is important for different projects in this space to be able identify points of interoperability. We consider this work as a concrete approach of collaboration in this community. We will continue to explore the points discussed in this paper and invite other projects to also work with us.

REFERENCES

[Aggarwal et al., 2004] Aggarwal R., Verma K., Miller J., and Milnor W., "Constraint Driven Web service Composition in METEOR-S," Proceedings of the 2004 IEEE International Conference on Services Computing (SCC 2004), Shanghai, China (September 2004) pp. 23-32.

[Cardoso et al., 2004] Cardoso J., Sheth A., et. al., "Quality of Service for Workflows and Web service Processes",Journal of Web Semantics, 2004

[Cimpian et al., 2005] E. Cimpian, M Moran, E. Oren, T. Vitvar, and M. Zaremba. Overview and Scope of WSMX. Technical report, WSMX Working Draft, http://www.wsmo.org/TR/d13/d13.0/v0.2/,February2005.

[de Bruijn et al., 2004] J. de Bruijn, D. Foxvog, K. Zimmerman: Ontology mediation patterns. SEKT Project Deliverable D4.3.1,Digital Enterprise Research Institute, University of Innsbruck, 2004.

[Keller et al., 2004] U. Keller, R. Lara, and A. Polleres, editors. WSMO Web service Discovery. WSMO Deliverable D5.1, WSMO Working Draft, 2004, latest version available at http://www.wsmo.org/2004/d5/d5.1/.

[METEOR] http://lsdis.cs.uga.edu/projects/past/meteor/

[METEOR-S] http://lsdis.cs.uga.edu/projects/METEOR-S/

[Rajasekaran et al., 2004] Rajasekaran P., Miller J., Verma K., Sheth A., "Enhancing Web services Description and Discovery to Facilitate Composition," Proceedings of the 1st International Workshop on Semantic Web services and Web Process Composition, July 2004

[Roman et al., 2005] D. Roman, J. Scicluna, Cristina Feier (eds.): Ontology-based Choreography and Orchestration of WSMO Services, WSMO deliverable D14 v0.1. available at http://www.wsmo.org/TR/d14/v0.1/, 2005.

[Sivashanmugam et al., 2003] Sivashanmugam K., Verma K., Sheth A., and Miller J., "Adding Semantics to Web services Standards", 1st International Conference on Web services, June 2003

[Sivashanmugam et al., 2004] Sivashanmugam K., Miller J., Sheth A., and Verma K., "Framework for Semantic Web Process Composition," International Journal of Electronic Commerce (IJEC), Special Issue on Semantic Web services and Their Role in Enterprise Application Integration and E-Commerce, Vol. 9, No. 2 (Winter 2004-5) pp. 71-106. M.E. Sharpe, Inc.

[SOA-RM]http://www.oasis-pen.org/committees/tc_home.php?wg_abbrev=soa-rm

[Verma et al., 2004] Verma K., Akkiraju R., Goodwin R., Doshi P., and Lee J., "On Accommodating Inter Service Dependencies in Web Process Flow Composition", 2004 AAAI Spring Symposium Series, March, 2004

[Verma et al., 2005] Verma K., Sivashanmugam K., Sheth A., Patil A., Oundhakar S., and Miller J., "METEOR-S WSDI: A Scalable Infrastructure of Registries for Semantic Publication and Discovery of Web services", Journal of

Information Technology and Management, Special Issue on Universal Global Integration, Vol. 6, No. 1 (2005) pp. 17-39. Kluwer Academic Publishers.

[Zaremba and Oren , 2005] Maciej Zaremba and E. Oren. WSMX Execution Semantics. Technical report, WSMX Working Draft, available at http://www.wsmo.org/TR/d13/d13.2/v0.2/, April 2005.

[Zaremba et al., 2005] Michal Zaremba, M. Moran, and T. Haselwanter. WSMXArchitecture. Technical report, WSMX Working Draft, http://www.wsmo.org/TR/d13/d13.4/v0.2/, April 2005.

[WS-CHOR] http://www.w3.org/2002/ws/chor/

[WSDL-S] Web service Semantics -- WSDL-S," A joint UGA-IBM Technical Note, version 1.0, April 18, 2005. http://lsdis.cs.uga.edu/library/download/WSDL-S-V1.pdf

[WSML, 2005] J. de Bruijn, H. Lausen, R. Krummenacher, A. Polleres, L. Predoiu, D. Fensel: The WSML Family of Representation Languages, WSML Working Draft v0.2, 2005, available at http://www.wsmo.org/TR/d16/d16.1/v0.2/

[XSLT] J. Clark (editor): XSL Transformations (XSLT), W3C Recommendation, 1999, available at http://www.w3.org/TR/1999/REC-xslt-19991116