



D13.3v0.1 Mediation

DERI Working Draft 26 May 2004

This version:

<http://www.wsmo.org/2004/d13/d13.3/v0.1/20040526>

Latest version:

<http://www.wsmo.org/2004/d13/d13.3/v0.1/>

Previous version:

<http://www.wsmo.org/2004/d13/d13.3/v0.1/20040426>

Editor:

Adrian Mocan

Authors:

Adrian Mocan
Emilia Cimpian
Eyal Oren
Matthew Moran
Michal Zaremba

This document is also available in non-normative [PDF](#) version.

Copyright ? 2004 [DERI](#)?, All Rights Reserved. [DERI](#) liability, trademark, document use, and software licensing rules apply.

Executive Summary

Mediation has a key role in WSMX execution environment, allowing the business partners to exchange integration messages. That is, each enterprise may use different problem domain ontology and it is necessary to provide a mechanism for making the integration and the communication between these enterprises possible. The aim of the mediation process associated with the execution environment is to mediate between two different ontologies, each of them used by a certain business partners.

The main assumption is that the involved ontologies conform to the WSMO specifications. Therefore in this deliverable we will develop a process that will deal with mediation between two WSMO conforming ontologies, by this being able to make full use of the advantage of having the same meta-level structural representation of both the source and the target ontology.

The ontology elements involved in the mediation process will be the concepts by means of their attributes and the non-functional properties. We will make use of the concepts internal structure for determining the elements part of the minimal common vocabulary shared by the two ontologies. The non functional properties will feed the information required to determine the lexical relations among those concepts that

offer no further information about their structure.

Our intentions are to provide solutions and tools for mapping a set of concepts from the source ontology with concepts from the target ontology. We will begin by analyzing a relatively small set of problems that may appear during the mediation process. Examples of these problems include different concept composition, incomplete concept representation on source (target) ontology, types mismatch, and more general or more specific concept definitions (subsumption). For each of the classes of analyzed problems we will provide a set of solution, starting from manual mediation solutions and moving towards semi-automatic solutions. In parallel we will develop a set of tools that will implement the presented strategies and will integrate the mediation component in WSMX. In the future the set of problems our component will be able to deal with will be extended for obtaining a mediation framework capable to cover a larger variety of mediation situations.

Table of contents

1. Introduction

2. Requirements

2.1. General requirements

2.2. Scope of mediation in the execution environment

2.3. Ontologies

2.4. Examples

2.5. Different composition of concepts

2.6. Incomplete/Over-complete concept representation

2.7. Data-type mismatch

2.8. Subsumption

3. Solutions

3.1. Decomposition

3.2. Different composition of concepts

3.3. Incomplete/Over-complete concept representation

3.4. Data-type mismatch

3.5. Subsumption

4. Component

5. Conclusions and further directions

References

Acknowledgments

1. Introduction

This deliverable presents our approach to mediation in the execution environment, describing both the problems we intend to address and the component designed to fulfill our aims. As described in [WSMX, 2004] the final scope of the execution environment is the domain defined by Web Service Modeling Ontology (WSMO) [WSMO, 2004] and the mediation component will have to address all forms of mediation specified in WSMO. But the scope of the WSMX in its first phase is WSMO-Lite [WSMO-Lite, 2004] and, as a consequence, the mediation in execution environment will start by addressing only the mediation case covered by WSMO-Lite: ontology to ontology mediation. Thus, we can say that, as a first phase, the

scope of this deliverable is data and information mediation.

It is well known now that models or ontologies describing the same or related problem domains are created by different entities around the world. This is the reason why more and more systems and applications require mediation for being able to integrate and use heterogeneous data sources. The mapping between models are required in several classes of application, as *Information integration and Semantic web*, *Data migration* or *Ontology Merging* [Madhavan et al., 2002].

Unfortunately it is very rare for a mapping tool to be able to automatically generate mapping rules. Some of the best results of research projects in this area are mapping tools that are able to validate or to suggest possible mappings but in some points of the mapping process the domain expert intervention still remains a necessity. Efforts for building such tools are concentrated in two main directions [Madhavan et al., 2002]: the first is the designing of heuristics based on structure or based on the non-functional properties (e.g. name) of the concepts involved, in order to generate the most plausible mappings [Noy & Musen, 2000]. Sometimes domain independent heuristics can also be used [Mitra et al., 2000]. The second approach is to use machine learning techniques for semi-automatically obtaining the required mappings. For example, [Doan et al., 2002] uses the instances of one ontology for learning a classifier for it, and then applies it for classifying the instances of the second ontology.

The mediation solution proposed by used is based on the first approach described above: we use decomposition methods for exploiting the internal structure characteristics of the concepts involved in the mediation process. In addition, strategies for suggesting the proper decompositions as well as the proper mappings between un-decomposable concepts will be provided. As from the second approach point of view, the machine learning based strategies are not suited for our needs because these methods require the usage of large sets of training data (in our case instances) for obtaining a high accuracy of the predicted mappings. But the aim of execution environment mediation is that for a given instance from the source ontology to provide the correspondent instance (or instances) from the target ontology. That is, no training data set is available but only the instance that has to be mediated.

The entire process of mediation described here happens on the schema level, all the mapping rules being created considering only schema information, without using any instance data. Furthermore, the fact that the execution environment scope is the WSMO domain implies that the mediated ontologies are conforming to WSMO specification, having the same schema specification. This means, for example, that a concept is defined in both source and target ontology following the same concept definition.

Although this deliverable describes the execution environment mediation, the entire process of mediation takes place outside the execution environment. The mediation component consists of two separate modules: the module responsible for generating the mapping rules and the module responsible for executing the mapping rules. The first module consists of a graphical environment that allows the expert domain to place his or her inputs and of the implementation of the required processes that assures the semi-automatic mediation behavior of the module. The second module offers an environment for executing the mapping rules. Its task is to receive the required mapping rules and the source instance(s), execute the mapping rule and deliver the corresponding target instance(s).

Section two will present the main problems the mediation module has to address, together with details related to the ontology elements that are considered in the mediation process. Also some examples will be introduced in this section which will help emphasize the main situations that this module will be able to handle.

In the third section we will present the solutions adopted for each of the situations described in the previous section.

Section four will describe the component that will handle the mediation problem in the execution environment.

In the end we will present some conclusions regarding our work as well as the further directions in which it can be extended.

2. Requirements

In the beginning of this section we will present in more details the role of the mediation component in the execution environment. After that we will briefly mention the ontology elements that we will consider in our approach, and in the end we will go through some examples and see a first classification of the problems that this component intends to address.

2.1. General Requirements

The mediation process consists of two main parts: the first part provides the mapping rules between a set of source concepts and a set of target concepts and the second part executes these mapping rules and returns the mediated data.

The creation of mapping rules is a process that in its incipient form is strongly based on human inputs (manual mapping) but it is also design to contain a set of strategies that creates the support for assisting the human during the entire process and by this, to transform the mapping process from a manual to a semiautomatic one. This process requires the usage of both the source and target ontologies for determining the definitions of the concepts that have to be mapped as well as for being able to determine all the other concepts involved in the initial definitions. In the end, as a result, it returns a mapping rule containing mappings among all the concepts involved in the initial concepts mappings.

The second part of the process has the task of executing the already generated mapping rules. This means that it has as inputs the source instance(s) together with the required mapping rule(s) and it provides as output the instance(s) of the target concept(s).

2.2. Scope of Mediation in the Execution Environment

Based on the above presented requirements we can define now the scope we aim to achieve through the mediation component which will be integrated in the execution environment. We see the mediation process in this context as being formed by two distinct steps, so, as a consequence, our mediation component is designed as consisting by two subcomponents. In the followings, the intended functionality for each of these two components is presented.

The first subcomponent, called *Mappings Maker*, has the role of creating the mappings between the two ontologies and the functionalities implied by this are:

- i. The capability of loading the source and target ontology and extracting the information that is relevant for the mediation process. In fact this component offers wrappers for reading the input ontologies and converting the relevant aspects in an internal representation.
- ii. The existence of graphical user interface that presents in a user-friendly manner the information affected by the mediation process. The GUI has three main sub-functionalities: provides the user the opportunity of inserting his/her inputs, displays the set of mapping suggestions and decompose the mapping problems in sub-problems to be solved individually.
- iii. Computes a set of suggestion based on the internal structure of the concepts to be mapped and on linguistic similarities of the name and other non-functional properties.
- iv. The possibility of saving the ontology relevant information and the already done mappings in an external storage for further usage. This functionality is defined, of course, in conjunction with the corresponding retrieving operations from the storage.
- v. The transformation of existing mappings in mapping rules conforming to the representation language of the input ontologies. For example if the input ontologies are represented in Flora the mapping rules will be generated as Flora rules. This functionality can also be part of the second subcomponent, depending if one desires the mapping rules to be generated at the runtime from already created mappings (in this case the place of this functionality is in the second subcomponent) or to be only executed at the runtime (the mapping rules are generated in this subcomponent and saved in the external storage).

The subcomponent presented above is not explicitly part of the WSMX architecture [[WSMX Architecture, 2004](#)], but enables the functionalities offered by the second subcomponent. The second subcomponent, called *Mediator*, is part of the WSMX architecture and offers the following functionalities:

- vi. The retrieval from the external storage of the mapping rules or of the mappings set, according with the chosen strategy (see v.). In the last case the rule creation functionality must be included in this subcomponent.
- vii. The creation of an environment for executing the mapping rules. For example in the case of Flora mapping rules a Flora environment able to execute Flora rules is provided.

In the first stage of WSMX and of the WSMX mediation module this component aims to solve only the set of problems described in the following sections. Part of the further work will be the extension of the problem set, and of course of the solutions proposed, together with the enhancement of the software component described above.

2.3. Ontologies

Web Service execution environment is meant to be the reference implementation for

Web Service Modeling Ontology [WSMO, 2004], and as a consequence the mediation process described in this deliverable refers to ontologies that conforms to [WSMO-Lite, 2004] specifications.

The execution environment (and by this the mediation component) makes no assumption regarding the owner or the scope of the target ontology (the one we try to mediated to). However, it is better to keep in mind that usually each of the business partners involved in common business processes are using different representations for their data sources. So trying to create and maintain different sets of mappings to all partners' data could be non-trivial at all, especially when the number of partners that want to communicate is growing. Our vision of mediation implies the existence of an agreed upon ontology that would reduce the total number of ontology to ontology mappings ([Omelayenko & Fensel, 2001], [Fensel & Bussler, 2002]). This is the reason why in the next section (the example section) one of the examples provided is considered to be the most complete and well structured from all the others, as being part of the above mentioned agreed upon ontology.

We mentioned before that the ontologies part of the mediation process are [WSMO-Lite, 2004] compliant. Our aim is to provide rules that specify all the required mappings between concepts from the source ontology and concepts from the target ontology. For creating these mappings we focus on attributes definitions that offer us the possibility of exploiting the internal structure of mediated concepts. Also the nonfunctional properties have a crucial role in creating the mappings between those concepts that don't offer any more details about their internal structure but these nonfunctional properties.

2.4. Examples

We analyze here three types of address format, each one of these being denoted by a different concept, and we will use F-logic syntax for their description. Using these examples we will identify the possible problems that can be encountered in the mediation process, due to their syntactic and semantic mismatches.

The first one, considered to be the most complete and well structured one, is described by a concept named *address0* having the following structure:

Listing 1: *Address0* address format example

```
address0[str => street,  
no => positive_int,  
loc => locality,  
county => county,  
country => country,  
zip => zip_code].  
  
positive_int[value => int].  
FORALL PI:positive_int <- PI>0.  
  
zip_code[value => string].
```

The second address format, denoted by *address1*, provides a different internal structure, and some of the attributes presented above as part of the *address0* definition are missing.

Listing 2: *Address1* address format example

```
address1[str_no => local,  
loc => locality,  
country => country,  
zip => zip_number].  
  
zip_number[value => integer].  
FORALL ZN:zip_number <- ZN>0.  
  
local[str => street,  
number => integer].
```

The last address format we are considering here, *address2*, is formed only by two string lines:

Listing 3: *Address3* address format example

```
address2[adr_line_1 => string,  
adr_line_2 => string].
```

The scope of the above examples is not to introduce the full set of problems that can appear during the mediation process. In fact, these examples have the role of helping to refer only the problems we address in our approach and the intention is to extend in the future this basic set of problems and enable our component to solve more complex situations.

2.5. Different Composition of Concepts

This is the most common problem that can appear during the mapping process and it is caused by the fact that semantically the same concepts are modeled in different ways using different internal descriptions. This problem occurs even from the beginning of mapping process and can be easily noticed in our examples too, no matter which source and target address format is chosen.

A particular case of this problem can be identified when different structures are used for modeling the same concept. That is, we can have both concepts referring to a set of semantically matching concepts (through their attributes) but using for example some intermediary concepts. This problem is also described as granularity of match in [Rahm & Bernstein, 2001] or simply as granularity in [Chalupsky, 2000]. Referring to our examples one can encounter this problem by mediating between *address0* and *address1* (or the other way around), when trying to map the attributes that denote *street* in both address formats (*local* is the intermediary concept).

2.6. Incomplete/Over-complete Concept Representation

Another possible problem in the process of mediation is the missing of the corresponding concept (either from the source or from the target ontology). In our examples, it is the case of the *county* which is not modeled in *address1*, although it appears in *address0*. For creating a mapping rule between these two concepts a decision has to be made regarding the missing concept and this decision mostly depends whether the missing concepts appears in the source or in the target. In the first case the data will be mediated without any content loss (even if from the target point of view the data will be incomplete), but in the second case some of the

content will be lost in the mediation process (but from the target point of view the data will be complete).

2.7. Data-Type Mismatch

Another common problem for the mediation process appears when concepts denoting the same aspects are defined using different data types. In our example it is the case of the *zip* attribute from *address1* and *address0* which has as a range *zip_number* and *zip_code* concepts, respectively. Both of them have only one attribute (*value*) but this attribute is *integer* in the first case and *string* in the second.

We could have different dimensions of this problem depending of the casting compatibility of the two data types involved. In our example when the mediation is done from the *address1* to *address0* the solution is very simple (integer can always be converted to a string) but if we consider the other way around the solution is not so trivial at all, since not any string can be converted to an integer.

2.8. Subsumption

When trying to mediate between concepts belonging to different ontologies the following situation may occur: some concepts from one of the ontologies are subsumed by one concept from the other ontology. In [Omelayenko & Fensel, 2001] this class of problems is referred as being 1:n, n:1 and n:m mapping cases and in [Rahm & Bernstein, 2001] it is considered to be part of the math cardinality process as set-oriented mapping case.

In our examples the mediation between *address0* and *address2* will lead to this kind of problem, i.e. it is easy to noticed that part of the *address0* attributes correspond to *adr_line_1* and the other part to the *adr_line_2* attribute in *address2*. In this case we can say that *Street*, *Locality*, *Country* concepts are simply subsumed by the *String* concept.

The subsumption problem complexity is dependent of where the subsumed concepts are placed: in the source ontology or in the target ontology. In the first case one can get from the subsumed concepts to the target concept by applying a certain operation (in our example could be concatenation). In the second case it is not so trivial to create a mapping rule that splits a given instance (with a no explicitly defined structure) in a set of subsumed concepts instances. But the most general case of subsumption implies the existence of the two previous cases together (n:m mapping) and, if referring to our examples, can be illustrated trying to mediate between two ontologies, both of them having defined the concept of address in the *address2* manner. Even if it could appear trivial to do the mappings in this case, we have to assume that each of the owners might create their instances in their own way and it is the role of the mediation process to do the required transformations.

3. Solutions

[Wiederhold, 1994] identifies three main approaches that deal with ontology integration. First of these approaches implies the use of domain experts for creating definitions for the terms in the used ontologies, definitions that must be accepted by all parties involved. When definitions are completed documented and released all the parties have to conform and adjust to these definitions. The second approach is

to assume that the terms from the subject ontologies are matching and further discovered mismatches are resolved by annotating the corresponding terms with some source or domain identifiers. The last approach, on a contrary, assumes that all the terms from the given ontologies are distinct and are denoting different concepts while it is not specified otherwise. This kind of information it is explicitly stated using mapping rules.

The first approach is used in [Dou et al., 2003] in the design of the OntoMerge system used for ontology merging and automated reasoning. The system uses a merged ontology (containing symbols and facts from the source and target ontology together with a set of bridging axioms) for performing translations between two ontologies. But this approach is not suited for our needs, our aim being to create a tool that provides the mapping rules in a semiautomatic manner and then executes them.

The second approach could be a useful one when knowing that the subject ontologies are using very similar terminology for modeling the problem domain. Unfortunately the ontologies are usually created independently so there could be major differences even in representing the same domain. So, the best choice for our mediation component remains the third one: all the concepts from the source and the target ontology are different until mapping rules specify otherwise.

The role of the mapping rules in the mediation process is to define how the information from the source maps to the target patterns. [Wache, 1999] considers that such mapping rules should provide two main functionalities: *combination* and *replacement*. The first functionality should be provided by a set of integration rules and the second one by context rules. The integration rules describes the target item in the terms of one source item and the context rules transform the information from the form required by the source context in the form required by the target context (e.g. currency transformations).

Our approach doesn't create such an explicit distinction between classes of rules used but the above mentioned functionalities are covered inside of each mapping rule. The way in which these aspects are addressed will be presented in the following sections.

In this section we will describe our approach and strategies used in the process of generating the required mapping rules for two given concepts (source concept and target concept) belonging to two different ontologies (source ontology and respectively target ontology). We will use the examples described in section 2.3. for illustrating these aspects in a more intuitive manner.

3.1. Decomposition

Since the ontologies describing the same real world domain are usually developed by different groups of domain experts, the compound concepts denoting the same thing could have different structure and composition. As a consequence for being able to provide correct mapping rules it is necessary to decompose both the source and the target concepts. [Lin et al., 2001] provides a set of translation rules that use the information available in a XML document and in its DTD for creating an Object Oriented Schema. The DTD rules are using the XML elements definitions for deciding if it is necessary to create types or functions (methods or attributes) in OO schema. In a similar manner we will make use of the internal structure of the

concepts to decide which strategy to apply further in the mapping process.

The decomposition of a concept consists of the identification of all its attributes and the creation of the mappings for each attribute range. One can easily notice that this is a recursive process that stops when a concept cannot be further decompose, i.e. the concept has no attributes defined. We named these concepts *primitive concepts*. The simplest examples of primitive concepts are *string* and *integer* but in our examples, beside this two, also *street* from *address0* can be consider a primitive concept because it is no further defined.

Another important remark about the decomposition is that it alternates the concept mappings with attribute mappings. That is, if the two concepts mapping requires decomposition, the next step will consist of a set of mappings between their attributes. But the mapping of two attributes always involves the mapping of their ranges which will lead again to concept to concept mapping.

The decomposition process may take place in only one of the concepts (source or target), but also in both of them. These possibilities will be emphasized in the next subsections when we will present the solutions for the problems described in section 2.

When finally no decomposition can be applied neither for the source nor for the target concept, we will find ourselves in the case of 1:1 mapping described in [Omelayenko & Fensel, 2001] or in [Rahm & Bernstein, 2001]. From here the process of mapping will continue with the analyze of the nonfunctional properties, for determining if the two concepts denote the same thing. In this case, the lexical relations [Wache et al., 2001] could be exploited together with the added information offered by the usage of nonfunctional properties.

3.2. Different Composition of Concepts

Usually this type of problem is addressed by applying the decomposition process both on the source concept and on the target concept. In fact this is the way in which the internal structure of a concept is revealed and the entire mapping is moving one level deeper involving now the concepts attributes (and of course their ranges) in the process of mediation. After the corresponding mappings are preformed the process returns to the upper level and proceeds with other mappings, if necessary.

For the particular case presented in section 2.4., when we have intermediary concepts involved, we will need to decompose only these intermediary concepts. In fact this is the case when the mappings involve primitive concepts on one side and decomposable concepts on the other side. In our examples it is the case of creating a mapping between the concepts denoting the street in both *address1* and *address0*. A decomposition of the local concept will be required, followed by a mapping between *local.str* attribute and *address0.str* attribute.

3.3. Incomplete/Over-complete Concept Representation

In general, for this kind of problems several solutions could be adopted, depending on the requirements of the two business partners.

When the missing concept is on the source side the mapping rule must assure that in the created instance (target instance) a default value is filled. Also there are some

cases when simply ignoring the missing concept could be enough, the responsibility of dealing with the missing value being delegated to the application that is using the new created instances.

When the missing concept is on the target side the problem is more complicated. Normally, the target doesn't know what to do with an instance of a concept that it cannot understand, but there may be some situations in which the target will have to return back that instance, for example, as a participation proof in the previous communication. So, a solution could be to simply discard, during the mapping rule execution, all the instances of an unmapped concept (assuming that it will be no need for these instances in the future) or mapping them as instances of a *bag* concept whose only role is to gather all unmapped instances.

3.4. Data-Type Mismatch

This problem can appear only in the final phase of the mediation process, when mapping between primitive concepts, after all the required decompositions were already done. At the attribute level the mappings can be simple assignments from the source attributes to the target attributes but also more complicated formulas that make the explicit conversion from one data format to the other. An example from our address formats mediation could be the mapping between *zip_code.value* and *zip_numer.value*. The checking that the mapping rule has specified the correct operation for conversing from one data format to the other is actually done only when the rule is executed. The operations performing the required transformations can be seen as the *context rules* described in [Wache, 1999].

3.5. Subsumption

The problem presented in the above section is a particular case of the subsumption problem from the point of view of the adopted solutions. That is, for this case we have to provide operations that are able to accept as input one or more source instances and to return one or more target instances. But after the second look, one can notice that this problem is far than being so easy to solve.

In fact, this operation can actually be easily provided if the subsumed concepts are in the source ontology, but if they are in the target ontology the problem becomes more difficult. We must mention here that this problem appears only when mapping between *primitive* concepts, so no other source concept (and implicitly instance) details are available. The only solution for this situation is to find out details about source instance's internal structure from other information sources (for example the instance provider).

In our examples this problem appears when mediating from *address2* to *address0*. The instance that fills the *add_line_1* attribute may contain the street, number, city or county information in any order. An operation that splits the content of this instance and creates the suited instances for the *address0* cannot be created without knowing more details about the source instance.

4. Component

The component responsible for mediation in the execution environment will consist of two main modules: the module that creates the mapping rules and the module

that allows the execution of the mapping rules.

Considering that a mapping rule becomes quite complicated even in the simple case of mapping *address0* to *address1*, and the fact that the human user decisions are still required in the mapping process, it is mandatory for the rules creator module to provide a graphical user interface. This user interface has several roles: to implement the decomposition process, to assist the user in his decisions, and to generate the required mapping rules. The second module will allow the loading of the suited mapping rules, their execution and the generation of the instances specified in the mapping rules.

5. Conclusions and Further Directions

This deliverable intends to provide a component that enables the mediation between concepts belonging to different ontologies. Having in the beginning an interface that requires an important human involvement and solution for a limited number of problems, this component will evolve in the future. It will provide a larger number of solutions for various mediation problems and will move from a manual mapping tool, to a semiautomatic one.

References


- [Chalupsky, 2000]** Hans Chalupsky. OntoMorph: a translation system for symbolic knowledge. In *Proceedings of 7th international conference on knowledge representation and reasoning (KR), Breckenridge, (CO US), pages 471–482, 2000.*
- [Doan et al., 2002]** A. Doan, J. Madhavan, P. Domingos, A. Halevy: Learning to map between Ontologies on the Semantic Web. In *WWW2002, 2002.*
- [Dou et al., 2003]** D. Dou, D. McDermott, P. Qi: Ontology Translation on the Semantic Web. In *ODBASE'03, 2003.*
- [Fensel & Bussler, 2002]** D. Fensel, C. Bussler. The web service modeling framework wsmf. In *White Paper and Internal Report Vrije Unversiteit Amsterdam, 2002.*
- [Lin et al., 2001]** H.Lin, T. Risch T. Katchaounov: “Adaptive data mediation over XML data”, To be published in *Journal of Applied System Studies (JASS)*, Cambridge International Science Publishing, 2001.
- [Madhavan et al., 2002]** Jayant Madhavan , Philip A. Bernstein , Pedro Domingos , Alon Y. Halevy: Representing and reasoning about mappings between domain models, *Eighteenth national conference on Artificial intelligence*, p.80-86, Edmonton, Alberta, Canada, July 28-August 01, 2002.
- [Mitra et al., 2000]** P. Mitra, G. Wiederhold, M. Kersten: A Graph-Oriented Model for Articulation of Ontology Interdependencies. In *Proceeding Extending DataBase Technologies, Lecture Notes in Computer Science*, vol. 1777. Springer, Berlin Heidelberg New York, 2000, pp. 86-100.
- [Noy & Musen, 2000]** N. Noy, M. Musen. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In *Proceedings of the National*

Conference on Artificial Intelligence (AAAI), 2000.

[Omelayenko & Fensel, 2001] B. Omelayenko, D. Fensel: An Analysis of Integration Problems of XML-Based Catalogues for B2B E-commerce. In *Proceedings of the 9th IFIP 2.6 Working Conference on Database (DS-9), Semantic Issues in e-commerce Systems*, Hong Kong, April 2001.

[Rahm & Bernstein, 2001] E. Rahm, P. Bernstein: A survey of approaches to automatic schema matching. *The VLDB Journal* 10: 334-350, 200.

[Wache, 1999] Holger Wache. Towards rule-based context transformation in mediators. In S. Conrad, W. Hasselbring, and G. Saake, editors, *International Workshop on Engineering Federated Information Systems (EFIS 99)*, Kuhlungsborn, Germany, 1999.

[Wache et al., 2001] H. Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, S. Hübner: Ontology-Based Integration of Information - A Survey of Existing Approaches. In *Proceedings of IJCAI-01 Workshop: Ontologies and Information sharing*, Vol. pp. 108-117, Seattle, WA, 2001. 

[Wiederhold, 1994] G. Wiederhold: Interoperation, Mediation, and Ontologies, In *Proceedings International Symposium on Fifth Generation Computer Systems (FGCS94), Workshop on heterogeneous Cooperative Knowledge-Bases*, Vol.W3, pages 33-48, Tokyo, Japan, Dec. 1994.

[WSMO-Lite, 2004] D. Roman, U. Keller, H. Lausen (eds.): *D11v02. Web Service Modeling Ontology - Lite (WSMO-Lite)*, version 0.2 available at <http://www.wsmo.org/2004/d11/v0.2/20040305/>

[WSMO, 2004] D. Roman, U. Keller, H. Lausen (eds.): *Web Service Modeling Ontology - Standard (WSMO - Standard)*, version 0.2 available at <http://www.wsmo.org/2004/d2/v0.2/index.html>

[WSMX, 2004] E. Oren, M. Zaremba, M. Moran: *D13.0v01 Overview and Scope of WSMX*, WSMO Working Draft v01, 2004.

[WSMX Architecture, 2004] M. Moran, M. Zaremba: *D13.4v01 WSMX Architecture*, WSMO Working Draft v01, 2004.

Acknowledgments

The work is funded by the European Commission under the projects DIP, Knowledge Web, Ontoweb, SEKT, and SWWS; by Science Foundation Ireland under the DERI-Lion project; and by the Austrian government under the CoOperate programm.

The editors would like to thank to all the members of the WSMO working group for their advices and inputs to this document.