



Data, Information and Process Integration  
with Semantic Web Services

## **DIP**

*Data, Information and Process Integration with Semantic Web Services*

**FP6 - 507483**

Deliverable

# **WP 7: Technology Watch and Standardization D7.4 Standardization Impact Analysis Update**

Joachim Quantz, Berlecon Research

Elmar Dorner, SAP

Christian de Sainte Marie, ILOG

Brahmananda Sapkota, NUIG

January 7<sup>th</sup>, 2004



## EXECUTIVE SUMMARY

This deliverable

- Gives an overview of the standards/formalisms used in DIP;
- Explains where exactly these standards/formalisms are used and what their purpose is in the context of usage;
- Discusses what alternatives could have been used and briefly analyzes the position of these alternatives in the market;
- Lists standards/formalisms still missing in DIP and discusses their potential use outside of DIP.

Summarizing the results,

- WSDL and SOAP are used as basic Web Services building blocks for describing and accessing component interfaces;
- WSMO/WSML is used as modeling framework for ontologies and service descriptions;
- RDF/OWL is still used by several partners for modeling due to the availability of tools for these standards – migration towards WSMO/WSML is envisaged as soon as tools based on these standards become available.

Due to the importance of RDF/OWL in the Semantic Web area and the wide range of tools supporting these standards it is strongly recommended to develop translators between RDF/OWL and WSMO/WSML in DIP. Such translators will allow the import/export of ontologies in different formats and will eliminate the format barrier between WSML and other ontology formal languages currently in use. They would thus contribute significantly to synergy effects between the W3C Semantic Web activity and the work on WSMO in the SDK cluster.

Disclaimer: The DIP Consortium is proprietary. There is no warranty for the accuracy or completeness of the information, text, graphics, links or other items contained within this material. This document represents the common view of the consortium and does not necessarily reflect the view of the individual partners.

## Document Information

<b>IST Project Number</b>	FP6 – 507483	<b>Acronym</b>	DIP
<b>Full title</b>	Data, Information, and Process Integration with Semantic Web Services		
<b>Project URL</b>	<a href="http://dip.semanticweb.org">http://dip.semanticweb.org</a>		
<b>Document URL</b>			
<b>EU Project officer</b>	Brian Macklin		




<b>Deliverable</b>	<b>Number</b>	7.4	<b>Title</b>	Standardization Impact Analysis Update
<b>Work package</b>	<b>Number</b>	7	<b>Title</b>	Technology Watch and Standardization








<b>Date of delivery</b>	<b>Contractual</b>	M 12	<b>Actual</b>	M 12
<b>Status</b>	Version 1.1		Final Version	
<b>Nature</b>	Prototype <input type="checkbox"/> Report <input checked="" type="checkbox"/> Dissemination <input type="checkbox"/>			
<b>Dissemination Level</b>	Public <input type="checkbox"/> Consortium <input checked="" type="checkbox"/>			



<b>Authors (Partner)</b>	Joachim Quantz (Berlecon Research), Elmar Dörner (SAP), Christian de Sainte Marie (ILOG), Brahmananda Sapkota (NUIG)			
<b>Responsible Author</b>	Joachim Quantz		<b>Email</b>	jq@berlecon.de
	<b>Partner</b>	Berlecon	<b>Phone</b>	+49 30 28 52 96 0

<b>Abstract (for dissemination)</b>	This deliverable gives an overview of the standards/formalisms used in DIP.	
<b>Keywords</b>	WSMO/WSML, OWL/RDF, SOAP, WSDL	

## Project Consortium Information

Partner	Acronym	Contact
National University of Ireland Galway	NUIG  National University of Ireland, Galway <i>Ollscoil na hÉireann, Gaillimh</i>	Prof. Dr. Christoph Bussler Digital Enterprise Research Institute (DERI) National University of Ireland, Galway Galway Ireland Email: <a href="mailto:chris.bussler@deri.org">chris.bussler@deri.org</a> Tel: +353 91 512460
Fundacion De La Innovacion.Bankinter	Bankinter 	Monica Martinez Montes Fundacion de la Innovacion. BankInter Paseo Castellana, 29 28046 Madrid, Spain Email: <a href="mailto:mmtnez@bankinter.es">mmtnez@bankinter.es</a> Tel: 916234238
Berlecon Research GmbH	Berlecon 	Dr. Thorsten Wichmann Berlecon Research GmbH Oranienburger Str. 32 10117 Berlin, Germany Email: <a href="mailto:tw@berlecon.de">tw@berlecon.de</a> Tel: +49 30 2852960
British Telecommunications Plc.	BT 	Dr John Davies BT Exact (Orion Floor 5 pp12) Adastral Park Martlesham Ipswich IP5 3RE, United Kingdom Email: <a href="mailto:john.nj.davies@bt.com">john.nj.davies@bt.com</a> Tel: +44 1473 609583
Swiss Federal Institute of Technology, Lausanne	EPFL 	Prof. Karl Aberer Distributed Information Systems Laboratory École Polytechnique Fédérale de Lausanne Bât. PSE-A 1015 Lausanne, Switzerland Email : <a href="mailto:Karl.Aberer@epfl.ch">Karl.Aberer@epfl.ch</a> Tel: +41 21 693 4679
Essex County Council	Essex 	Mary Rowlett, Essex County Council PO Box 11, County Hall, Duke Street Chelmsford, Essex, CM1 1LX United Kingdom. Email: <a href="mailto:maryr@essexcc.gov.uk">maryr@essexcc.gov.uk</a> Tel: +44 (0)1245 436524
Forschungszentrum Informatik	FZI 	Andreas Abecker Forschungszentrum Informatik Haid-und-Neu Strasse 10-14 76131 Karlsruhe Germany Email: <a href="mailto:abecker@fzi.de">abecker@fzi.de</a> Tel: +49 721 9654 0

<p>Institut für Informatik, Leopold-Franzens Universität Innsbruck</p>	<p>UIBK            Universität          innsbruck</p>	<p>Prof. Dieter Fensel          Institute of computer science          University of Innsbruck          Technikerstr. 25          A-6020 Innsbruck, Austria          Email: <a href="mailto:dieter.fensel@deri.org">dieter.fensel@deri.org</a>          Tel: +43 512 5076485</p>
<p>ILOG SA</p>	<p>ILOG            Changing the rules of business</p>	<p>Christian de Sainte Marie          9 Rue de Verdun, 94253          Gentilly, France          Email: <a href="mailto:cma@ilog.fr">cma@ilog.fr</a>          Tel: +33 1 49082981</p>
<p>inubit AG</p>	<p>Inubit            the integration experts</p>	<p>Torsten Schmale          inubit AG          Lützowstraße 105-106          D-10785 Berlin          Germany          Email: <a href="mailto:ts@inubit.com">ts@inubit.com</a>          Tel: +49 30726112 0</p>
<p>Intelligent Software Components, S.A.</p>	<p>iSOCO  </p>	<p>Dr. V. Richard Benjamins, Director R&amp;D          Intelligent Software Components, S.A.          Pedro de Valdivia 10          28006 Madrid, Spain          Email: <a href="mailto:rbenjamins@isoco.com">rbenjamins@isoco.com</a>          Tel. +34 913 349 797</p>
<p>The Open University</p>	<p>OU  </p>	<p>Dr. John Domingue          Knowledge Media Institute          The Open University, Walton Hall          Milton Keynes, MK7 6AA          United Kingdom          Email: <a href="mailto:j.b.domingue@open.ac.uk">j.b.domingue@open.ac.uk</a>          Tel.: +44 1908 655014</p>
<p>SAP AG</p>	<p>SAP  </p>	<p>Dr. Elmar Dörner          SAP Research, CEC Karlsruhe          SAP AG          Vincenz-Priessnitz-Str. 1          76131 Karlsruhe, Germany          Email: <a href="mailto:elmar.dorner@sap.com">elmar.dorner@sap.com</a>          Tel: +49 721 6902 31</p>
<p>Sirma AI Ltd.</p>	<p>Sirma    <b>Ontotext</b>          Knowledge and Language          Engineering Lab of Sirma</p>	<p>Atanas Kiryakov,          Ontotext Lab, - Sirma AI EAD          Office Express IT Centre, 3rd Floor          135 Tzarigradsko Chausse          Sofia 1784, Bulgaria          Email: <a href="mailto:atanas.kiryakov@sirma.bg">atanas.kiryakov@sirma.bg</a>          Tel.: +359 2 9768 303</p>
<p>Tiscali Österreich GmbH</p>	<p>Tiscali  </p>	<p>Dieter Haacker          Tiscali Österreich GmbH.          Diefenbachgasse 35          A-1150 Vienna          Austria          Email: <a href="mailto:Dieter.Haacker@at.tiscali.com">Dieter.Haacker@at.tiscali.com</a></p>

		Tel: +43 1 899 33 160
Unicorn Solution Ltd.	<p>Unicorn</p> 	<p>Jeff Eisenberg            Unicorn Solutions Ltd,            Malcha Technology Park 1            Jerusalem 96951            Israel            Email: <a href="mailto:Jeff.Eisenberg@unicorn.com">Jeff.Eisenberg@unicorn.com</a>            Tel.: +972 2 6491111</p>
Vrije Universiteit Brussel	<p>VUB</p> 	<p>Carlo Wouters            Starlab- VUB            Vrije Universiteit Brussel            Pleinlaan 2, G-10            1050 Brussel ,Belgium            Email: <a href="mailto:carlo.wouters@vub.ac.be">carlo.wouters@vub.ac.be</a>            Tel.: +32 (0) 2 629 3719</p>

---

**TABLE OF CONTENTS**

<a href="#"><u>EXECUTIVE SUMMARY</u></a> .....	II
<a href="#"><u>TABLE OF CONTENTS</u></a> .....	VII
<a href="#"><u>1 INTRODUCTION</u></a> .....	1
<a href="#"><u>2 STANDARDS AND FORMALISMS USED IN DIP</u></a> .....	1
<a href="#"><u>2.1 A technology map of standards and the scope of DIP</u></a> .....	1
<a href="#"><u>2.2 Summary of Standards' Usage in DIP</u></a> .....	3
<a href="#"><u>3 STANDARDS AND FORMALISMS STILL MISSING IN DIP</u></a> .....	3
<a href="#"><u>3.1 Current Status</u></a> .....	3
<a href="#"><u>3.2 Analysis</u></a> .....	4
<a href="#"><u>3.3 Some Requirements on a Policy Representation Standard</u></a> .....	5
<a href="#"><u>3.3.1 Policy Enforcement and Production Rules</u></a> .....	5
<a href="#"><u>3.3.2 Reasoning about policies and inference rules</u></a> .....	6
<a href="#"><u>3.4 A proposed standardisation strategy for policy representation</u></a> .....	6
<a href="#"><u>3.4.1 A layered approach</u></a> .....	6
<a href="#"><u>3.4.2 A stack of XML-based languages for representing policies inDIP</u></a> .....	7
<a href="#"><u>4 CONCLUSION AND RECOMMENDATIONS</u></a> .....	8
<a href="#"><u>5 APPENDIX</u></a> .....	9
<a href="#"><u>5.1 WSMO</u></a> .....	9
<a href="#"><u>5.1.1 Motivation for using WSMO in DIP</u></a> .....	10
<a href="#"><u>5.1.2 Usage of WSMO in DIP</u></a> .....	11
<a href="#"><u>5.1.3 Potential Alternatives to WSMO</u></a> .....	12
<a href="#"><u>5.2 OWL/RDF</u></a> .....	12
<a href="#"><u>5.2.1 Motivation for using OWL/RDF in DIP</u></a> .....	12
<a href="#"><u>5.2.2 Usage of OWL/RDF in DIP</u></a> .....	12
<a href="#"><u>5.2.3 Potential Alternatives to OWL/RDF</u></a> .....	13
<a href="#"><u>5.3 WSDL</u></a> .....	13
<a href="#"><u>5.3.1 Motivation for using WSDL in DIP</u></a> .....	13
<a href="#"><u>5.3.2 Usage of WSDL in DIP</u></a> .....	13
<a href="#"><u>5.3.3 Potential Alternatives to WSDL</u></a> .....	14
<a href="#"><u>5.4 SOAP</u></a> .....	14
<a href="#"><u>5.4.1 Motivation for using SOAP in DIP</u></a> .....	14
<a href="#"><u>5.4.2 Usage of SOAP in DIP</u></a> .....	14
<a href="#"><u>5.4.3 Potential Alternatives to SOAP</u></a> .....	15
<a href="#"><u>5.5 UDDI</u></a> .....	15



---

<a href="#">5.5.1 Motivation for using UDDI in DIP</a> .....	15
<a href="#">5.5.2 Usage of UDDI in DIP</a> .....	15
<a href="#">5.5.3 Potential Alternatives to UDDI</a> .....	15
<b><a href="#">REFERENCES</a></b> .....	<b>16</b>

## 1 INTRODUCTION

This deliverable gives an overview of standards and formalisms used in DIP.<sup>1</sup> It provides a snapshot of the current state within DIP and will be updated every 6 months during the remainder of the project.

Section 2 presents a technology map of standards and the scope of DIP and briefly describes the standards and formalisms currently used in DIP.

Section 3 is concerned with standards that are or that will be required for the success of DIP and, more generally, Semantic Web Services, and that are still missing.

Section 4 concludes with recommendations for further action.

The Appendix contains more detailed material on the standards/formalisms discussed in Section 2, namely:

- WSMO/WSML/WSMX
- OWL/RDF
- WSDL
- SOAP
- UDDI

For each standard, the following information is provided:

- Motivation of using the standard in DIP
- Description of usage in DIP
- Brief discussion of potential alternatives and reasons why they are not used

## 2 STANDARDS AND FORMALISMS USED IN DIP

### 2.1 A technology map of standards and the scope of DIP

Figure 1 shows a (simplified) technology map of meta-models, languages and protocols for the Semantic Web Services. It presents a functional view on which existing standard stacks can be easily mapped: as an example, such a mapping is shown in the bottom-right corner of the components for the W3C and some other proposed specifications in the Web Service and semantic Web stacks<sup>2</sup>.

For our purpose, we subdivided the picture in three columns:

- the main column, labelled “description and representation”, in the center, concerns the languages and meta-models for describing and representing the

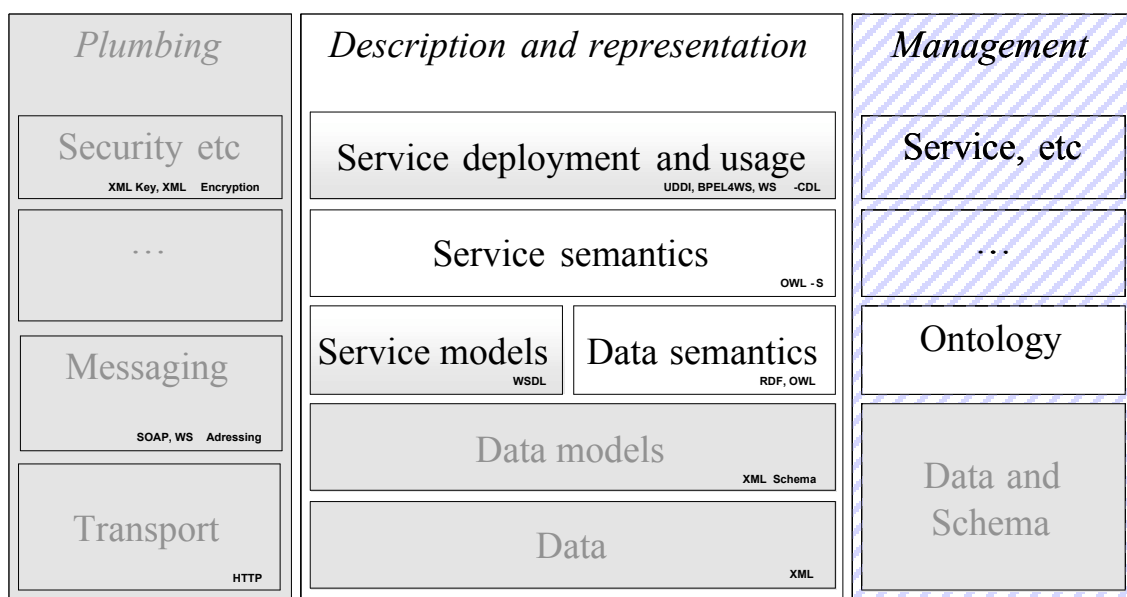
---

<sup>1</sup> We here use the terms “standards” and “formalisms” in a very wide sense, as not all of the languages and formats discussed in this deliverable might be considered standards or formalisms in a strict sense.

<sup>2</sup> This representation of the technology map and the example mapping are meant to be illustrative of our purpose rather than complete and exhaustive. Notice further that, although we tried to make the layering as meaningful as possible, there is no necessary or exact correspondence between the same level components in different columns.

various objects (or aspects of these objects) that are published, shared or exchanged in a Semantic Web Services environment, from data and data models, to ontologies, service models and up;

- the left-most column, labelled “plumbing” concerns the meta-model and languages required for moving these objects around in an efficient and secure way, from the transport layer up to the security and trust aspects etc.;
- the right-most column, labelled “management”, concerns the meta-models and languages required for managing them. At first sight, that column might seem less relevant with respect to standardisation. However, management may, for instance, require specific information to be attached to the managed objects (and thus transported along with them), e.g. for tracking purposes, and that information has to be standardised.



**Figure 1: Semantic Web Services standards technology map**

DIP focuses on description and representation languages and meta-models, and more specifically on the middle to upper layers of the stack. Notice that being in the scope of DIP does not mean that a language or meta-model is isolated or relevant to Semantic Web Services technology only: indeed, DIP builds directly on Web Services and Semantic Web technology.

The shadings and patterns describe the status of the components represented on the map as seen from the DIP project:

- Shaded components are not specific to Semantic Web Services.
  - Completely shaded components are inherited from the general Web and they are not specifically impacted by the Web being semantic or not, as far as DIP is concerned (e.g. HTTP for transport, XML for data exchange, XML Schema for data model exchange). They are thus out of the scope of DIP;
  - Components with the shaded lower half belong to the regular, “non-semantic” Web Services stack as well, but they are extended for

Semantic Web Services: this is of course especially the case for the service modelling and the service deployment and usage components. They are in the scope of DIP, but, on these subjects, DIP must build on widely accepted legacy specifications and/or collaborate with communities outside the realm of Semantic Web Services.

- Patterned components are depending on work in progress. For that reason, they are excluded from the current of scope of DIP; however, the relevant ones might be included in DIP's scope in the future, when the technology on which they depend will be mature enough. Most of the management is contained within that scenario, with the noticeable exception of ontology management;
- Plain components are either specific to Semantic Web Services (e.g. service semantics) or deeply impacted. They are therefore fully in the scope of DIP. Moreover, they are areas where DIP can and should take a position of leadership.

## 2.2 Summary of Standards' Usage in DIP

This section briefly summarizes the usage of standards and formalisms in DIP. More detailed information on the standards and formalisms discussed here can be found in the Appendix below.

- WSDL and SOAP are used as basic Web Services building blocks for describing and accessing component interfaces. They are used by DIP components offering service functionality. In doing so, the Basic Profile of WS-I (Web Services Interoperability Organisation) will be taken into account [33]. More details on WSDL and SOAP are presented in Appendix 5.3 and 5.4.
- WSMO/WSML has been chosen as the backbone of the DIP project and is used as modeling framework for ontologies and service descriptions. It is used as the basis for all technical components to be developed in DIP and providing functionality covered by WSMO/WSML. More details on WSMO/WSML are presented in Appendix 5.1.
- RDF/OWL is still used by several partners for modeling due to the availability of tools for these standards – migration towards WSMO/WSML is envisaged as soon as tools based on these standards are available. Due to the widespread usage of RDF/OWL and their status of being official recommendations by the W3C, some DIP tools will also support import/export functionality for RDF/OWL. More details on RDF/OWL are presented in Appendix 5.2.
- UDDI is considered as a potential basis for the DIP registry. No final decision has been taken yet, however, as it might be more straightforward to base such a semantic registry directly on an ontology tool supporting WSMO and/or OWL/RDF. More details on UDDI are presented in Appendix 5.5.

## 3 STANDARDS AND FORMALISMS STILL MISSING IN DIP

### 3.1 Current Status

A survey of the scientists, architects and developers working on the technical work packages and the use cases did not reveal the existence of an acknowledged need for

additional standards besides the ones currently in use – or whose use is planned at a later stage – or under development in the project.

The general agreement is that, all the requirements, which are not covered by existing standards or work in progress in the relevant standardisation organisations, are under development or planned within the WSMO framework. WSMO/WSML will be proposed as a standard to the appropriate organisation at some point, and thus it satisfies the requirement for the formats, protocols etc used in the project to be accepted standards. However, we believe that a layer dedicated to policies is missing from the protocol stack.

### 3.2 Analysis

The technology map shown in Figure 1 and discussed in Section 2.1 helps explain the negative result of the enquiry: all the components in DIP's focus are indeed covered by existing standards or work in progress (including WSMO). However, it also reveals the relative immaturity of a domain that still focuses mostly on the functional issues and does not yet take the practical, business-like use, usage and usability aspects into account.

Practically, the world of Web services is not uniform: even at the technical level, the same function can be achieved in several ways, and a client and service may support one or more of these ways or specific combinations. As a consequence, clients and services have either to agree off-line on how the service will be provided (technically), or they have to expose their (technical) constraints and capabilities.

The diversity of the technical eco-system in which Web services strive is now widely recognised and it starts being taken into account in the technical architecture of Web services: see for instance the work on WS-Policy [25], WSPL [26] or the organisation by the W3C of a workshop on Constraints and Capabilities for Web Services<sup>3</sup> [24].

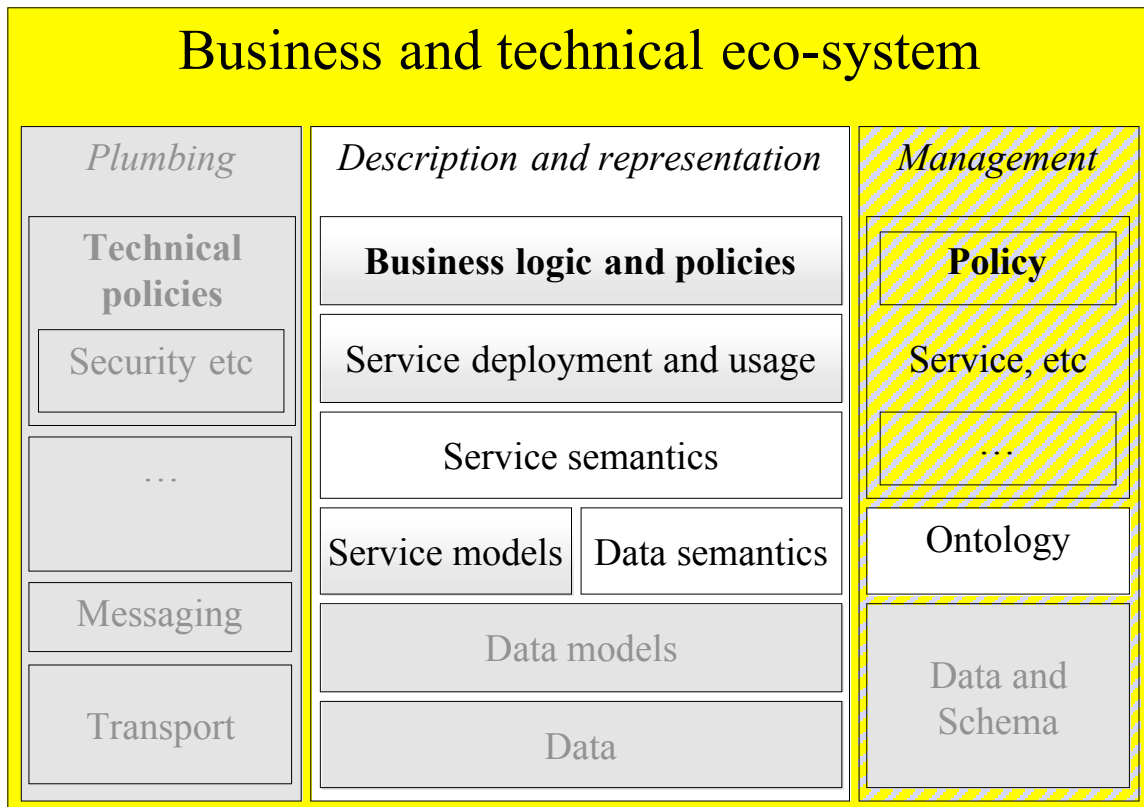
However, the fact that the success of Web services will ultimately depend on how well they cope in a highly dynamic and diverse business and regulatory eco-system has still to be recognized and dealt with. In the real world (that is, in a world of users and businesses etc; as opposed to the pure world of technology), the delivery of information, the access to services, and the execution of processes are usually subject to all kinds of policies, rules and regulations. In a Web Service environment, the enforcement of these policies, rules and regulations has to be automated.

If Web Services are to be composed, relevant policies as well as applicable rules and regulations must be composed as well: they have thus to be exposed. Semantic Web Services up the stake, as it loosens the coupling further: The alternative to exposing the relevant policies, rules and regulations along with a published Web Service is to check the policy and regulatory compatibility beforehand using non-automated means!

In other words, a necessary step in the advance of Web services and Semantic Web Services towards technical and business maturity is the explicit exposure of all the relevant policies, rules and regulations, and, as a consequence, the addition of an explicit and separate policy layer in the standards stack (see Figure 2).

---

<sup>3</sup> Note that two position papers submitted by DIP partners were accepted for the workshop ([27], [28]) and that their authors attended the workshop.



**Figure 2: Extended SWS standards technology map**

Although the inclusion of CPP/A in the ebXML architecture shows the right intuition, ebXML does not actually deal with business policies nor even acknowledge them.

### 3.3 Some Requirements on a Policy Representation Standard

#### 3.3.1 Policy Enforcement and Production Rules

Policies<sup>4</sup> are atomic, highly specific and structured statements that constrain some aspect of a business process or activity that controls or influences its behaviour, or, usually, highly structured sets of such statements.

Specifically, policies are rules or sets of rules that state under which conditions certain actions can or have to be executed. The enforcement of a policy is defined as the systematic execution of the associated actions whenever the condition is satisfied.

It is not always obvious that a policy implies an action, especially when it is expressed as a constraint or a capability: this is because the guarded actions can be implicit in the context. As a consequence, policies are often confused with the condition of the rules that they state. However, a policy without an associated action would be void, as it could not be enforced.

For instance, if a Web Service publishes that “a client **MUST** encrypt a specific header with WS-Security using a X.509 or user name security token”, the associated action,

<sup>4</sup> Including regulations, business rules etc. From now on, we will use the term ‘policy’ in the broader sense that covers both local policies (e.g. technical constraints and capabilities, business rules and policies) as well as non-local ones (e.g. regulations).

when the condition is satisfied, is to deliver the service. Another example would be a regulation that requires that, “users of the service **MUST** be of legal age”: the guarded action, here again, is the delivery of the service. A classical example of a business policy where the associated action is explicit is when “a gold customer gets a 10% discount”.

A unit policy statement is thus made of the description of a condition and the, possibly implicit, specification of an action or set of actions. In other words, they are production rules according to the definition accepted in Artificial Intelligence and Cognitive Psychology, and a standard for the representation of policies must satisfy the requirements for representing production rules.

### **3.3.2 Reasoning about policies and inference rules**

It is generally accepted that the description of the conditions in policy statements are represented by logical expressions for machine processing. The expression that represents the condition of a unit policy statement can be used as a pattern to recognise the situations where the policy applies, e.g. using a pattern-matching algorithm or a querying mechanism: this is typically what a production rule system does.

It can also be used as a logical assertion as part of a reasoning, e.g. using an inference engine. In the second example, above, for instance, a reasoner could use the knowledge that “only legal age people can have a credit card”, and the fact that the service requester provided a valid credit card number as part of his profile, to infer that the requester is of legal age and that the service can thus be delivered.

More generally, policies often contain definitions from which parts of the enforcement conditions must be inferred: in our example above, the business policy that defines the discount to be given to gold customers is likely to contain also the definition of the different classes of customers (e.g. in terms of their buying history). That definition must be used to infer the status of a particular customer, and to determine whether the condition for a discount is met.

In addition, when negotiation is allowed, policies define modes of enforcement, such as “must”, “may”, “should”, “preferred” etc. Modal inference can then be used to reason about the policies, e.g. to find a compromise policy acceptable to the Web Services participating in a composed service, or to both a client and a service. As a consequence, a standard for the representation of policies must satisfy the requirements for representing inference rules as well, including modal inference.

## **3.4 A proposed standardisation strategy for policy representation**

### **3.4.1 A layered approach**

As a consequence of this double set of requirements, a layered approach to the standardisation of policy representation should be preferred. We propose to sub-divide the policy layer into three sub-layers:

- A common base layer for the representation or modelling of logical expressions. With respect to policies, that layer regards only the standardisation of the condition part. However, the scope of that common base extends beyond the specific need of policy sharing and enforcement, to cover the requirements for reasoning (including modal reasoning) and logical inference as well. That

common base would thus provide a natural bridge between (non-semantic) Web Service technology and Semantic Web applications;

- A middle layer for the representation or modelling of the operational semantics associated to policy expressions. The differences between the requirements for representing production rules and requirements for representing inference rules are dealt with at that layer, though a family of standards.

The standardisation of the constructs that enable the unambiguous modelling or representation of how policy statements are combined with respect to enforcement, belong to that layer, as well as any constructs that are specific to the enforcement of policies. Deeper examination might reveal that several standards may be the best way to serve different classes of policy enforcement platforms. However, the mainstream approach to policy enforcement, that is, Business Rules engines, must be dealt with.

Any constructs that would be specific to logical inference, or to a class of logical inference platforms, would also belong to that layer, as a way to combine logical statements. We saw that reasoning about policies was part of making Web services really semantic. However, these constructs should be a separate set from the ones needed to deal with policy enforcement.

Finally, application specific enforcement mechanisms probably concern vertical application specific standards covering all three layers. Such standards should however be anchored to the common base layer in order to be open to usage extension.

- An upper layer for the application specific aspects: the particular terms that populate a policy statement do not belong to application specific vocabularies (which can be standards by themselves), along with the specification of the possible actions.

### 3.4.2 A stack of XML-based languages for representing policies inDIP

Figure 3, below, represents the corresponding stack of XML-based languages for publishing and exchanging policies<sup>5</sup>:

- The base layer should be a standard XML-based language for representing and exchanging logical expressions. It should leverage the existing proposals that include such a language (such as RuleML, SWRL, SRML) as well as related query languages (e.g. XML Query), and it should build on the expertise and experience from RDF, RDF Schema, RDF Query, OWL, WSML Rules, REI [30], etc.

XML-based standards for representing policies in vertical domains exist or are currently under development, such as WS-Policy. Since such standards are not based on the logical language under discussion here, for obvious reasons, the specific constructs they use for the description of the condition part of policies must be mapped onto that language. That should not be a problem, since these

---

<sup>5</sup> The colouring conventions are the same as for figures 2 and 3: completely shaded means out of scope for DIP; partly shaded means in scope for Semantic Web Services and DIP, but in the scope of (that is, requiring cooperation with) other communities as well; plain means fully in scope for Semantic Web Services or even DIP-specific.

vertical standards can be expected to be either quite restrictive (e.g. WS-Policy has only two constructs for that purpose: <ExactlyOne> and <All>, which are semantically equivalent to the logical ‘and’ and ‘exclusive or’) or conceptually close to our logical language (e.g. the description language defined in WSPL) ;

- The middle layer must contain an XML-based language for representing production rules. That language should focus on the operational semantics and the action part, that is, on enabling the unambiguous specification of the execution of rules and rule sets. It must also be compatible (in an XMI sense) with the Production Rules Representation meta-model currently under development at OMG. If further examination reveals the existence of other policy enforcement paradigms that are not reducible to or compatible with the production rules model, the (different) XML-based languages enabling the unambiguous representation of their operational semantics would also belong to that layer.

If required, specific markup needed by inference engines or classes of inference engines belong to that layer too, as one or several different languages. That would include the parts of such XML-based languages as RuleML and SWRL that are not included in the common logical language or mapped onto its constructs.

Markup specific to the operational semantics of policies in vertical domain can be expected to be included in vertical policy standards (as is the case e.g. in WS-Policy). This does not preclude it from being mapped onto the constructs in the standards for the appropriate generic enforcement platforms (e.g. business rules engine);

- With the possible exception of the specification of the actions attached to policies, we do not expect application-specific vocabularies beyond the XML schema for their object- or data-model to require any specific markup for the description of policies.

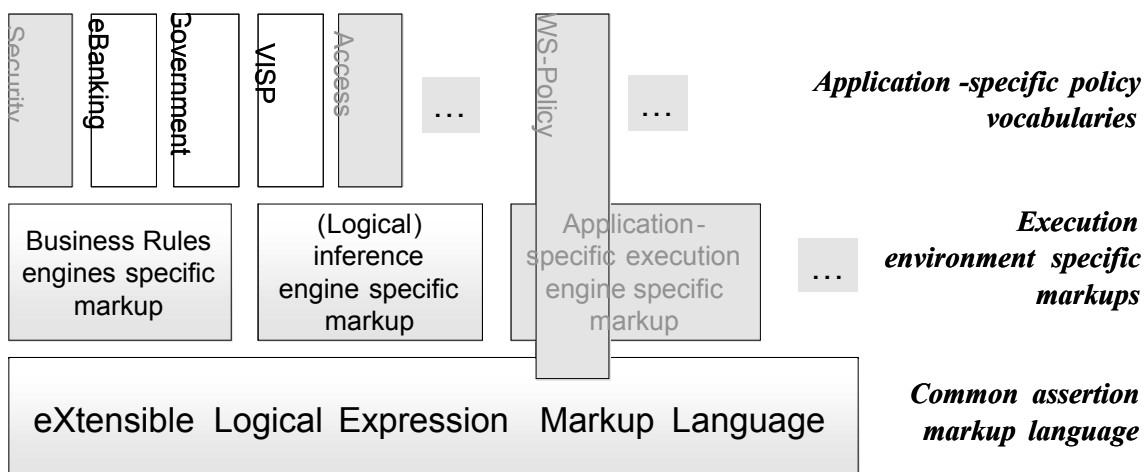


Figure 3: A proposed stack for policy markup

#### 4 CONCLUSION AND RECOMMENDATIONS

Summarizing the results,

- WSDL and SOAP are used as basic Web Services building blocks for describing and accessing component interfaces;
- WSMO/WSML is used as modeling framework for ontologies and service descriptions;
- RDF/OWL is still used by several partners for modeling due to the availability of tools for these standards – migration towards WSMO/WSML is envisaged as soon as tools based on these standards are available.
- Although no requirement for the development of new standards has been expressed within the project, the analysis of the current standards stack reveals that a policy layer is missing and that the management issues may not be taken enough into account.

It is strongly recommended to develop translators between OWL and WSML. On the one hand, this will help users to migrate existing ontologies in OWL format to WSML. On the other hand, it would allow testing WSMO with existing ontology tools. In general, such translators would guarantee that WSMO/WSML is not an isolated development. They would thus contribute significantly to synergy effects between the W3C Semantic Web activity and the work on WSMO in the SDK cluster.

On the same subject, it is strongly recommended that DIP take an active or even leading role in the emergence of a Semantic Web Services activity in the W3C.

It is also strongly recommended that DIP supports and participates in the development of a standard or a set of standards enabling the publication and exploitation of policies, rules and regulations. DIP should take an active or even leading role to make that development happen and to support the proposed standardization strategy (cf. Section 3.4).

With respect to the extension of standards to take into account the management issues, or the definition of new standards for that purpose, it is recommended that the requirements of the use cases in DIP be scrutinized and that the emergence of effort outside DIP be monitored. The need for further action should be re-examined periodically.

## 5 APPENDIX

This appendix contains more detailed material on the standards/formalisms summarized in Section 2.2.

### 5.1 WSMO

The Web Service Modelling Ontology (WSMO) [3] is an ontology that defines a set of concepts necessary to model the main aspects of Semantic Web Services. It aims to address the problems of existing Web Services by semantically enriching them to allow automatic discovery, composition, interoperation and invocation. It is conceptually based on the Web Service Modelling Framework (WSMF) [4] and adheres to the principles of loose coupling and strong mediation services. WSMO is led by the Semantic Web Services working group of the SDK cluster<sup>6</sup>.

WSMO concentrates on four main components defined by WSMF; namely web services, ontologies, goals and mediators, to define concepts related to Semantic Web Services. The underlying description language of WSMO is Web Service Modelling

---

<sup>6</sup> <http://sdk.semanticweb.org>

Language (WSML). WSML represents a family of languages covering description logic, first-order logic and logic programming. The expressed level of computability and complexity varies from one member of the WSML family to another.

Detailed information on WSMO and WSML can be found in [3] and [9] respectively.

### **5.1.1 Motivation for using WSMO in DIP**

In the following, we list the main characteristics of WSMO explaining why it has been chosen as the backbone formalism in DIP.

#### **Expressiveness**

Expressiveness of the description language used in a Web Service is crucial in order to present its capabilities clearly. The WSMO description language WSML provides layers of languages each having a different level of logical expressivity and computability. Having these layers of expressivity users can choose between the computability and expressivity of a language according to the requirements of a particular application domain.

The family of WSML languages is layered as follows: WSML-Core consists of the intersection between description logic and logic programming, WSML-DL extends this in the direction of an expressive description logic, WSML-Flight and WSML-Rule extend WSML-Core in the direction of logic programming, and WSML-Full unifies both branches in a first-order logic with non-monotonic extensions. These languages offer increasing expressivity but come with decreasing complexity results [9].

#### **Clarity**

WSMO defines various components of a Web Services clearly separating the concerns of interests. Choreography, for example, is defined as a concept in WSMO in order to handle multiple request-response cycles between a service requester and a service provider that might occur while executing a service request. Defining these components explicitly simplifies the integration process since it provides flexibility in choosing appropriate mediators. Similarly, WSMO clearly decouples the goals, and the services and ontologies are defined to ensure their applicability in a specific application domain. Having axiomization in WSMO/WSML facilitates the definition of relations between the requester's input and the outputs of a service.

#### **Flexibility**

WSMO supports extendable non-functional properties. As WSMO allows its mediators to add missing conditions, service requester and service provider can be defined without assuming that the inputs, outputs, preconditions and effects (IOPEs) of a service profile are subset of those defined in the service model [7].

#### **Executability**

WSMO aims at providing a set of execution semantics and reference implementations. There is already a Web Service Execution Environment (WSMX) version v1 available under SourceForge<sup>7</sup>. WSMX is an open source implementation. It adopts loosely coupled architecture patterns and provides a means to describe all the aspects of the Semantic Web Services.

---

<sup>7</sup> <http://www.sourceforge.net>

In addition to the aforementioned characteristics, WSMO is based on a well defined conceptual framework WSMF and provides the components such as globally defined non-functional properties, a meta-ontology, goals, mediators, choreographies. WSMO goals can be reused using ggMediators to refine existing (pre-defined) goals. It allows importing ontologies from other domains, as WSMO provides conceptualization of domain ontologies. It also provides means to compensate *errors* that might occur while using a service. To define temporal and casual relationships of multiple message exchanged, the message exchange pattern is defined in WSMO. Since Web Services interact via mediators only it provides a base for integration [13].

### 5.1.2 Usage of WSMO in DIP

Considering the vision of DIP, the usage of WSMO has a great impact in achieving the goals and objectives of DIP. DIP will receive contribution from WSMO [8] in different work packages from different perspectives as outlined below.

#### Language/Ontology:

By providing inputs on “Language Neutral API”, WSMO can contribute to *WP2 – Ontology Management*. WSMO is used as a basis to define various business protocol and process ontologies in *WP3 – Service Ontologies and Service Description*. Similarly, the WSMO repository can be used as goal repository for DIP as specified under the same work package. It can also be used as a service description framework. Service ontologies and Service description integration can be achieved based on WSMO, WSML and WSMX [8]. Likewise, WSMO mediator concepts can be used in *WP4 – Service Mediation* to integrate heterogeneous data, process and information in order to ensure different services within or outside companies[EAWK1]. *WP5 – Service Usage*, on the other hand, can use WSMO service discovery and service invocation concepts in order to enable the users to use the service.

#### Prototype:

In *WP2*, it can contribute through WSML to develop a prototype that is used to browse and version a web service modelling ontology.

#### Architecture:

*WP6 – Interoperability and Architecture*, can use the WSMO conceptual model and the architecture style of WSMX. The WSMO conceptual model provides maximum flexibility in terms of interoperability by providing different types of mediators. WSMX architecture inspires the loosely coupled pattern thus providing maximum extendibility. Therefore, the DIP architecture can be developed following the principles adopted in WSMX.

#### Publicity:

*WP14 – Training*, WSMO tutorials and training materials can be used [8].

#### Case studies:

In *WP6*, WSMX can be used as an execution environment for executing different case studies that are under consideration in DIP.

### 5.1.3 Potential Alternatives to WSMO

One of the alternatives of WSMO is OWL-S which provides an upper ontology to describe a Web Service. OWL-S service profiles are linked to Web Services using the *presents* property, which does not have cardinality restrictions allowing zero or more profiles of a service [2]. It assumes that both goals and service capabilities can be described using common global ontologies, which is not realistic because service providers and service requesters can have different levels of desired expressivity and computability. The lack of mediators in OWL-S makes it difficult to reuse and/or to import ontologies. It is not clear how OWL-S handles a request/response cycle that might occur while executing a service. Because of the lack of a conceptual framework it is difficult to understand the meaning of some of the defined concepts and their relationships [31]. Similarly, OWL-S poorly supports inferences as it does not provide rich axiomization [13]. [32] talks about an OWL-S virtual machine as an execution environment but this is not yet publicly available.

## 5.2 OWL/RDF

RDF [1] and OWL [2] are two recommendations of the Semantic Web Activity at the W3C. The Resource Description Framework (RDF) is a language for representing information about resources in the World Wide Web. RDF is intended for situations in which this information needs to be processed by applications, rather than being only displayed to people. RDF provides a common framework for expressing this information so it can be exchanged between applications without loss of meaning. OWL builds on RDF and RDF Schema and adds more vocabulary for describing properties and classes: among others, relations between classes (e.g. disjointness), cardinality (e.g. "exactly one"), equality, richer typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes.

### 5.2.1 Motivation for using OWL/RDF in DIP

Although WSMO has been chosen as the ontology standard for DIP, OWL and RDF are also used in the project. There are two main reasons for this. On the one hand, tools for OWL/RDF are already available, whereas WSMO is a new standard and tools are still under development. Thus, several partners are currently using ontologies based on OWL and/or RDF in their initial implementations. These will then be migrated towards WSMO.

The second reason for supporting OWL/RDF is that these are standards recommended by the W3C and hence widely used. Providing adapters to import/export OWL/RDF in DIP tools allows users of these technologies to migrate towards WSMO, facilitating the pick-up of this new standard considerably. It will also allow the mixed usage of WSMO and OWL/RDF and thus contribute to the openness of the tools developed in DIP.

### 5.2.2 Usage of OWL/RDF in DIP

The Business Data Ontology developed in WP3 (D3.3) has been built in OWL Lite. The main reason for this choice was the availability of ontology tools for this language. Conceptually, the model is compatible with WSML Core thus guaranteeing that a migration to WSML is straightforward, once WSML-based ontology tools are available.

The KAON-2 reasoner developed in WP1 supports both WSML Core/FLite and OWL-DL. It will be used as a basis for discovery in WP4.

The Publishing tool developed in WP4 will initially support OWL/RDF and underlying RDF stores such as Jena or SESAME. Again, the long term strategy is to use WSMO-based tools for publishing once these are available and to use OWL/RDF-based tools in the meantime to gather practical experiences with initial implementations.

The financial ontology in WP10 has been built with the OWL-plugin of Protégé. It will be migrated to WSMO as soon as a WSMO-based ontology tool is available. The ontology is lightweight, does not contain any axioms, and has been designed with WSMO-compatibility in mind. The migration will thus be straightforward.

### **5.2.3 Potential Alternatives to OWL/RDF**

As has been explained above, WSMO can be seen as an alternative to OWL/RDF and it is expected that partners currently using OWL/RDF will migrate to WSMO in the course of the project. Another alternative would be to use proprietary formats based on plain XML. This, however, would fail to leverage the potential provided by existing Semantic Web technology, e.g. with respect to tools for storage, inferences, or queries.

## **5.3 WSDL**

WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information.<sup>8</sup> The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate, however, the only bindings described in this document describe how to use WSDL in conjunction with SOAP 1.1, HTTP GET/POST, and MIME. [21]

### **5.3.1 Motivation for using WSDL in DIP**

Web Services Description Language (WSDL) is a specification to describe networked XML-based services. It provides a simple way for service providers to describe the basic format of requests to their systems regardless of the underlying protocol (such as Simple Object Access Protocol or XML) or encoding (such as Multipurpose Internet Messaging Extensions). WSDL is a key part of the effort of the Universal Description, Discovery and Integration (UDDI) initiative to provide directories and descriptions of such on-line services for electronic business. [20]

WSDL plays an important role in the overall Web services architecture since it describes the complete contract for application communication (similar to the role of IDL in the DCOM architecture). Although other techniques exist for describing Web services, the WS-I Basic Profile Version 1.0 mandates the use of WSDL and XML Schema (see Figure 4) for describing Web services. This helps ensure interoperability at the service description layer. [23, 29]

### **5.3.2 Usage of WSDL in DIP**

All technical oriented work packages in DIP have chosen to describe their interfaces by using WSDL and taking into account the WS-I Basic Profile [33]. The DIP architecture

---

<sup>8</sup> For more information on how WSDL and SOAP work together see [34,35,36].

is assembled with the components build in these work packages. All three DIP case studies are building on the general DIP architecture therefore they also using WSDL for the description of their service interfaces.

### **5.3.3 Potential Alternatives to WSDL**

WSDL 1.1 is considered the de-facto standard today because of it's industry-wide support. Most Web services toolkits support WSDL 1.1, but there have been some interoperability problems across the different implementations. Many developers believe that the extensive flexibility of WSDL (and the resulting complexity) is the fundamental source of these problems. The WS-I has helped resolve some of these issues by encouraging developers to use certain parts of the specification and discouraging them from using others. The W3C is actively working on the next "official" version of WSDL: 2.0, WSDL 1.2 was dropped after some time, but it's currently only a Working Draft and not supported by the mainstream toolkits, if any [23]. It is not likely that during the remaining project time of DIP a shift towards the updated version of WSDL is necessary.

## **5.4 SOAP**

SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls and responses. SOAP can potentially be used in combination with a variety of other protocols; however, the only bindings defined in this document describe how to use SOAP in combination with HTTP and HTTP Extension Framework. [22]

### **5.4.1 Motivation for using SOAP in DIP**

SOAP is rapidly becoming the generally accepted protocol for XML- based system-to-system communication. Because several important Web services infrastructures rely on SOAP for XML message transfer, many developers have adopted SOAP as an essential Web service tool.

At best, SOAP introduces a level of indirection to such XML message exchanges by embedding an XML message in a SOAP envelope. Since the SOAP envelope can carry metadata about the original XML message, such as processing instructions, the envelope can aid a Web service in processing that message. At worst, SOAP makes it difficult, if not impossible, to verify the validity of an XML message traversing between two Web services.

SOAP is needed when a Web service is designed, not as a series of message exchanges, but as a Web-accessible API. SOAP, and its various implementations, automate much of the marshalling and unmarshalling of method parameters and return values when invoking that API. [17]

### **5.4.2 Usage of SOAP in DIP**

As SOAP is dealing with the exchange of messages, this is not an issue for the work packages WP1-3. Currently for the technical oriented workpackages WP 4-6 and also the DIP case studies WP 8-10 SOAP is the choice of offering for using Web services through standardized interfaces. E.g. the service invocation component in DIP WP4

uses SOAP for message transfer and WSDL for the description of the interface. WS-I Basic Profile [33] will be taken into account when using SOAP in DIP.

### **5.4.3 Potential Alternatives to SOAP**

SOAP is not required when offering XML based Web services in order to conform to the W3C definition of Web services [18,19]. For many Web services, you can go with a simpler combination of XML, HTTP, and an application-specific message protocol.

Beside this technical possibility for not using SOAP there is a stronger, but more business driven, reason for using SOAP: In the light of interoperability between business processes and the emerging trend to offer Web Services as a business model, many companies choose SOAP as their way to implement access to Web services.

To summarize, there may be a simpler way or alternatives to offer Web services but for the sake of interoperability, currently SOAP is the way to go. This is especially true for the project DIP whose main objective is to provide an open architecture with exploitable tools.

## **5.5 UDDI**

UDDI [16] creates a standard interoperable platform that enables companies and applications to quickly, easily, and dynamically find and use Web services over the Internet. UDDI also allows operational registries to be maintained for different purposes in different contexts. UDDI is a cross-industry effort driven by major platform and software providers, as well as marketplace operators and e-business leaders within the OASIS standards consortium.

### **5.5.1 Motivation for using UDDI in DIP**

UDDI is the registry standard used by traditional Web Services and is thus a natural candidate for the DIP registry. However, UDDI is lacking support for semantic descriptions so far. In the OASIS Technical Committee responsible for UDDI there is some discussion on a "Semantic UDDI", e.g. by adding an "rdfBag" to UDDI. Since it is not clear yet, how exactly such a Semantic UDDI will look like and to which degree a UDDI-style registry is actually needed, such a registry is currently not yet implemented in DIP. It will be decided during the second project year whether the implementation of such a registry will be undertaken or not.

### **5.5.2 Usage of UDDI in DIP**

As described in detail in D4.2 (Publishing Process Specification), UDDI can be used as the underlying registry for publishing and discovery of Semantic Web Services. It is not yet clear, however, whether this approach will be actually implemented in DIP or whether DIP will only support a registry based on an ontology repository (see section below). In any case, the DIP architecture is designed flexible enough to support both types of registries.

### **5.5.3 Potential Alternatives to UDDI**

Instead of basing the DIP registry on UDDI, an approach based on an ontology repository is much more in line with the general philosophy underlying DIP. Using an ontology repository as the basis of the registry allows fully leveraging the potential of semantic descriptions. For one thing, service descriptions can be easily combined with the domain model, since both use the same representation language. Moreover,

inference engines can be integrated into the repository to check consistency and derive additional information from the available data. Finally, a generic query language for WSMML and/or RDF can be used for querying the registry.

## REFERENCES

- [1] <http://www.w3.org/RDF/>
- [2] <http://www.w3.org/2004/OWL/>
- [3] Roman D.; Lausen H.; Keller U. (2004): *WSMO-Standard*. WSMO Working draft, version 0.2.
- [4] Fensel, D.; Bussler, C.; Ding, Y.; Omelayenko, B. (2002a): *The Web Service Modeling Framework WSMF*. Electronic Commerce Research and Applications, 1(2), 2002.
- [5] <http://www.wsmo.org/wsmx/>
- [6] Arroyo S.; Bussler C.; Kopeck\_ J.; Lara R.; Polleres A.; Zaremba M. (2004): *Web Service Capabilities and Constraints in WSMO*.
- [7] Lara R.; Roman D.; Polleres A.; Fensel D. (2004): *A conceptual comparison of WSMO and OWL-S*. European Conference on Web Services, Erfurt, Germany, 2004.
- [8] Vasiliu L.; Moran M.; Bussler C.; Roman D. (2004): *WSMO in DIP*, WSMO Working Draft, version 0.2
- [9] de Bruijn J.; Lausen H.; Fensel D.(2004): *The WSML Family of Representation Languages*, WSML Working Draft, version 0.2
- [10] The OWL Services Coalition: OWL-S: Semantic Markup for Web Services
- [11] Sycara K.; Martin D.; McGuinness D.L.; McIlraith S.; Paulucci M.: *OWL-S Technology for Representing Constraints and Capabilities of Web Services*, On behalf of the OWL-S Coalition, In collaboration with Tim Finin, Grit Denker, Lalana Kagal.
- [12] Christensen E.; Curbera F.; Meredith G.; Weeravarana S. (2001): *Web Service Description Language (WSDL) 1.1*, W3C Note.
- [13] Balzer S.; Liebig T.; Wagner M (2004): *Pitfalls of OWL-S – A practical Semantic Web Use Case*, ICSOC 2004, November 15-19, New York.
- [14] Bussler C.; Arroyo S.; Stollberg M.; Moran M.; Domingue J.; Zaremba M.; Cabral L.; de Bruijn J.(2004): *WSMO-Tutorial*, Net Objects Day 2004, Erfurt, Germany, 17 September
- [15] Bussler C.; Arroyo S.; Stollberg M.; Moran M.; Domingue J.; Zaremba M.; Cabral L.; de Bruijn J.(2004): *WSMO-Tutorial*, The Eleventh International Conference on Artificial Intelligence: Methodology, Systems, Applications, The Semantic Web Challenge, Varna, Bulgaria, September 2<sup>nd</sup>-4<sup>th</sup>.
- [16] Universal Description, Discover and Integration (UDDI). <http://www.uddi.org/>
- [17] Sommer, F.: *Why use SOAP?* Artima developer. <http://www.artima.com/webservices/articles/whysoap.html>, March 2003.
- [18] W3C: *SOAP Version 1.2 Part 0: Primer*. <http://www.w3.org/TR/soap12-part0/>, W3C Recommendation 24 June 2003.

- 
- [19] W3C: *Web Services Glossary*. <http://www.w3.org/TR/2002/WD-ws-gloss-20021114/>, W3C Working Group Note 11 February 2004.
- [20] IBM: *Using WSDL in SOAP applications*. <http://www-106.ibm.com/developerworks/webservices/library/ws-soap/?dwzone=ws>, Nov. 2000.
- [21] W3C: *Web Services Description Language (WSDL) 1.1*. <http://www.w3.org/TR/wsdl>, W3C Note 15 March 2001.
- [22] W3C: *Simple Object Access Protocol (SOAP) 1.1*. <http://www.w3.org/TR/2000/NOTE-SOAP-20000508>, W3C Note 08 May 2000.
- [23] Aaron Skonnard, Northface University: *Understanding WSDL*. <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnsoap/html/argsoape.asp>, Microsoft Library, October 2003.
- [24] <http://www.w3.org/2004/09/ws-cc-program>
- [25] D. Box, et al, "Web Services Policy Framework (WS-Policy)," September 2004, available at <http://msdn.microsoft.com/ws/2004/09/Policy/>
- [26] "XACML profile for Web Services", working draft 4, September 03; available at <http://www.oasis-open.org/committees/download.php/3661/draft-xacml-wspl-04.pdf>
- [27] *Web Service Capabilities and Constraints in WSMO*, from Sinuhé Arroyo, Christoph Bussler, Jacek Kopeck\_, Rubén Lara, Axel Polleres, Michal Zaremba. Digital Enterprise Research Institute (DERI). Available at <http://www.w3.org/2004/08/ws-cc/wsmo-20040903>
- [28] *Why we need an XML standard for representing Business Rules*, Christian de Sainte Marie. ILOG. Available at <http://www.w3.org/2004/08/ws-cc/cdsm-20040903>
- [29] W3C: *Web Services Description Language (WSDL) 2.0*, <http://www.w3.org/TR/wsdl20/>, W3C Working Draft 3 August 2004.
- [30] *Declarative Policies for Describing Web Service Capabilities and Constraints*, Lalana Kagal, Tim Finin, and Anupam Joshi, position paper for the W3C workshop on Constraints and Capabilities for Web Services. Available at <http://www.w3.org/2004/08/ws-cc/umbc-20040904>
- [31] Mika P., Oberle D., Gangemi A., Sabou M.:(2004): *Foundations for Service Ontologies: Aligning OWL-S to DOLCE*, In the proceedings of WWW 2004, May 17 -22, 2004, New York.
- [32] Paolucci M., Ankolekar A., Srinivasan N., and Sycara K.: *The DAML-S Virtual Machine*. In *2nd International Semantic Web Conference (ISWC2003)*, September 2003.
- [33] W S - I            B a s i c            P r o f i l e            <http://www.ws-i.org/deliverables/workinggroup.aspx?wg= basicprofile>.
- [34] <http://www.xml.com/pub/a/2002/02/20/endpoints.html>
- [35] <http://www.brics.dk/~amoeller/WWW/webservices/index.html>
- [36] <http://www.developer.com/services/article.php/1602051>
-