3rd International Conference on Complex Distributed Systems (CODS 2009)
The Second SIWN Congress (SIWN 2009), Leipzig, Germany, 23-25 March 2009

Declarative Data Grounding Using a Mapping Language *

Miguel García, José María Álvarez, Diego Berrueta and Luis Polo

R&D Department, Fundación CTIC
Parque Científico Tecnológico, 33203, Gijón, Asturias, Spain
Email: {miguelg.rodriguez,josem.alvarez,diego.berrueta,luis.polo}@fundacionctic.org

Abstract: Grounding is the process in charge of linking requests and responses of web services with the semantic web services execution platform, and it is the key activity to automate their execution in a real business environment. In this paper, the authors propose a practical solution for data grounding. On the one hand, we define a mapping language to relate data structures from services definition in WSDL documents to concepts, properties and instances of a business domain. On the other hand, two functions that perform the lowering and lifting processes using these mapping specifications are also defined.

Keywords: service-oriented architectures, web services, semantic webs, ontologies, data grounding, semantic web services, mapping languages

1. Introduction

Web services are the base technology for Service Oriented Architectures (SOA) on the Web. According to the architecture and definitions by W3C [5], a web service is a software system designed to support interoperable machine-to-machine interaction on the net. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

However, practical deployment of SOA architectures usually faces problems of integration due to the heterogeneity of the services, in particular, integration of different data models. Semantic Web Services (SWS) [1, 3, 13, 23] combine current web services and the semantic web technology [2, 29]. More specifically, SWS propose ontologies as common data models to abstract the definition of services. Consequently two different levels of service description appear [7, 8]: some tasks can be automated at the semantic level (e.g. discovery of services) while others can only be performed combining the semantic and syntactic descriptions of the services (e.g. invocation). The latter require a mechanism to translate data between this two levels.

Data Grounding is the bidirectional process that downgrades a semantic model to a syntactic level through a subprocess called *lowering* and upgrades a syntactic model to a semantic level through a subprocess called *lifting*, enabling actual invocation of web services in SWS environments, see Fig. 1.

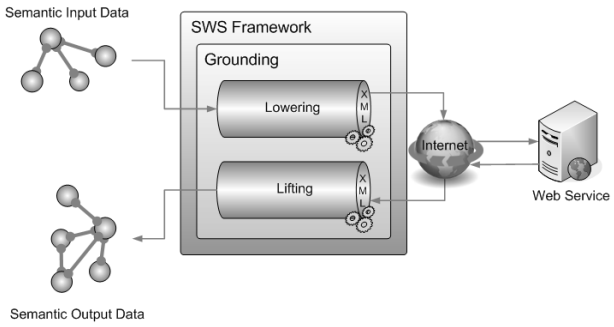

Figure 1. Grounding Process

We focus on data grounding [19] because it is the cornerstone to deploy semantic web services in production environments. Data grounding allows to build a request from the information available in the semantic model and to process the response from the service. In this paper, we present a new approach to data grounding based on a declarative mapping language. We address the problem from a structural point of view, i.e., independently from the logical foundations of ontologies. Moreover, our proposal is restricted to the vertical transformations between different description levels of services. Horizontal transformations such as data mediation are assumed to be solved by other components of the SWS platforms, therefore they are out of the scope of this paper.

This paper is structured as follows. Firstly, previous related work is reviewed in the next section and we highlight our main contributions in Section 3. Afterwards, Section 4 provides auxiliary definitions required to present our proposal in Section 5. Next section introduces a short summary about the project supporting this work. Our proposal is compared to other grounding approaches in Section 7. Finally, last section provides the main conclusions and future work.

2. Related work

Taking into account the current web technological stack, the problem of data grounding in semantic web services consists of a structural transformation between XML trees (SOAP Messages) and RDF [16] graphs (ontological data). Trees are special kind of graphs, therefore algebraic approaches to graph transformation, such as the double-pushout (DPO) and the single-pushout (SPO), can be used [11]. Although these techniques are a feasible solution from the formal point of view

* This work is part of PRAVIA project, partially funded by the Principality of Asturias, cod. IE05-172, and developed in conjunction with E2000 Nuevas Tecnologías

they are too complex from a practical point of view. The particular structure of XML trees and RDF graphs make it possible to devise specific solutions.

According to [29] there are three approaches to data grounding, see Fig. 2:

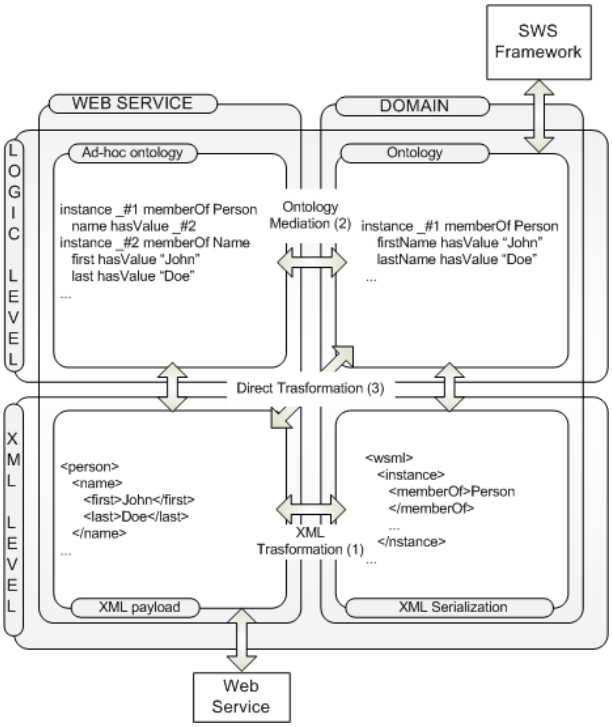

Figure 2. Approaches to Data Grounding. The upper half represents data in WSML ontologies.

1. Transformations at the XML level are the first option to travel between a semantic model (serialized as XML) and a syntactic representation. The main advantage is that this process is based on existing and robust technologies: XSLT or XQuery, besides the serialization of a semantic model to XML is simple, *a priori* (e.g. RDF/XML serialization of a RDF graph). Nevertheless, there is a lack of homogeneity. Existing ontology languages like RDF, OWL or WSML-DL, etc. are based on graphs and they have different ways to be serialized as XML. Mainstream research is focused on this approach to implement grounding [17, 28], e.g. WSMO initiative [18, 22].
2. Transforming at the ontology level involves building an *ad-hoc* pseudo-ontology from the XML Schema of WSDL [4]. The grounding is implemented using ontology mediation [24] between that pseudo-ontology and the domain ontology. This method is based on merging, aligning and mapping techniques [6]. The mappings are used as a set of rules that are executed on a rule engine, e.g. FLORA-2[1]. The key point of this approach lies on the reliability and expressiveness of the built pseudo-ontology. Anyway, tools available to realize this approach are still in an early stage of maturity.
3. Direct transformations between elements from the XML document to entities belonging to a domain. There are two possible approaches: (a) *ad-hoc* transformation code that must be manually programmed in a language such as Java or XSPARQL [20]; (b) using declarative mappings rules between XML Schema and the semantic model. In the second case, the mapping rules are written in a particular language and it is necessary to implement a processor to perform the transformations. In this field, SAWSDL [15, 21] and its predecessor WSDL-S [1] are examples of this approach, but the implementations are not yet available or are committed to a particular project. Alternatively, OWL-S extends both WSDL and OWL to specify grounding.

All described approaches require human intervention at design and validation time. In the first two options the human intervention is more complex than in the third one. In the first case, transformations are coded into programs, in the second one mapping rules are declared via rules in complex representational languages [25, 27]. Therefore we finally chose the third option because it is more suitable for people not trained in programming or knowledge engineering. It only needs human intervention to directly map entities with XML elements at design time but knowledge about technologies (e.g. XSLT or Flora2), operations (e.g. merging or aligning) or algorithms (e.g. PROMPT or GLUE) are not required.

3. Main contributions

Our approach for *data grounding* is based on a **direct mapping language** $\mathcal{M}$ between the syntactic description of the messages of a web service (XML Schema [14] inside WSDL) and the semantic model built with ontology languages like RDFS(S), OWL or WSML. These mappings are clear instructions to transform between the two different description levels. Such transformations take place when a semantic agent needs to exchange information with a web service. We propose two functions to: 1. generate XML content for SOAP requests and 2. interpret XML content of SOAP responses to create a graph.

4. Preliminary concepts

This section reviews some concepts and technologies from the two stacks: syntactic and semantic, see Fig. 3. On the one hand, the **Syntactic Level** contains the technologies and concepts related to web services from a syntactic point of view. On the other hand, the **Semantic Level** is built with the semantic technologies and concepts expected to enable dynamic and scalable cooperation between different systems and organizations.

	Syntactic Level	Semantic Level
Data Structure	Tree ↔ (Trasformation, Run Time)	Graph
Description of Structures	RTG ↔ (Mapping, Design Time)	Ontology
Path	Sequence of elements	Sequence of labelled edges
Identifier	QNAME	IRI

Figure 3. Relation between syntactic and semantic representations.

[1] It is a logic-based object-oriented language (that relies on the XSB inference engine) for building knowledge-intensive applications. It is based on F-logic, HiLog and Transaction Logic. http://flora.sourceforge.net/

Data Structure. At the syntactic level, XML Information Set [12] provides the abstract model to refer to the information in an XML document. An information set can contain up to eleven different types of information items. However, only *Document Information Item*, *Element Information Item* and *Character Information Item*s are necessary to define our mapping language and functions.

At the semantic level, the data model is the RDF one, i.e., an unordered set of triples $(s, p, o) \in (I \cup B) \times I \times (I \cup L \cup B)$, where I is the universe of all possible named resources, identified by a IRI, B is an infinite set of pairwise different, unnamed resources, and L is the set of all RDF literals. A set of RDF triples is structurally interpreted as a graph $\mathcal{G} = (V, E)$, where V is a set of vertex, $V \subseteq (I \cup B \cup L)$, and E is a set of edges, $E \subseteq V \times I \times V$. For each triple (s, p, o) of the RDF data model there is a directed edge, labeled p, between vertices s and o.

Description of data structure. At the data structure level, XML documents are trees and their structure is described using XML Schema. The latter is a language to define regular tree grammars (RTG) from the theory of formal languages [10, 26]. This kind of grammars make it possible to see XML documents as trees obtained by derivation using the production rules. In the following, we interpret XML documents as trees. A tree grammar $G = (NT, \Sigma, S, R)$ is composed of a set NT of non-terminal symbols, a set Σ of terminal symbols, an initial symbol S with $S \subset NT$ and a set R of production rules of the form $\alpha \longrightarrow \beta$ where α, β are trees of $T(\Sigma \cup NT \cup X)$ and X is a set of variables, α contains at least one non-terminal. Moreover we require that $\Sigma \cap X = \emptyset$ and $NT \cap X = \emptyset$. More specifically, a RTG, $G = (NT, \Sigma, S, R)$ is a tree grammar such that all non-terminal symbols have arity 0 and production rules have the form $A \longrightarrow \beta$, with A a non-terminal symbol of NT and β a tree of $T(\Sigma \cup NT)$.

At the semantic level, two major SWS frameworks have been proposed: WSMO [13] and OWL-S [23]. These SWS frameworks describe services capabilities, i.e. *what* services do, and their interfaces, i.e. *how* to communicate with them by external agents. Additionally, SWS frameworks are extended with domain ontologies to describe the entities that take part of the *precondition*, *postcondition*, *effect* and *assumptions* of a semantic web service description. More specifically, a particular request or response of a web service is a dataset $\mathcal{D}$ described using an ontology $\mathcal{O}$.

Path. A path in a XML tree is the sequence of nodes from a source node (*Document Information Item* or *Element Information Item*) to a target node (*Element Information Item* or *Character Information Item*).

In the graph world, let $\alpha = \{p_1, p_2, \ldots, p_n\}$, where $p_i \in I$, be a sequence of edge labels in a RDF graph $\mathcal{G}$ that defines a path in $\mathcal{G}$. We define the interpretation of a path, $[\![\alpha]\!]^{\mathcal{G}}(v)$, as the set of vertex reached from the source vertex v following path α. This interpretation function can be computed recursively:

$$
\begin{aligned}
[\![\emptyset]\!]^{\mathcal{G}}(v) = &\quad \{v\} \\
[\![\{p_1, p_2, \ldots, p_n\}]\!]^{\mathcal{G}}(v) = &\quad \bigcup \{[\![\{p_2, \ldots, p_n\}]\!]^{\mathcal{G}}(v') \\
&\quad /(v, p_1, v') \in E\}
\end{aligned}
$$

Identifiers. QNames and IRIs are unique identifiers in the XML and RDF realms respectively.

4.1 Relation between WSDL and RTG

XML Schema is used in WSDL documents to define the structure and elements of valid messages exchanged with the web service. These messages are the input and output of the operations allowed by a web service. The payload of SOAP messages is built (in the request of the service) and parsed (in the response of the service) according to the XML Schema grammar defined in the WSDL document. We assume that XML Schema is used for describing web services messages following the best practices in order to improve practical interoperatiblity [9]. Thus these messages are valid, well-formed and interoperable according to the formal definition of the web service.

The RTG $G = (NT, \Sigma, S, R)$ of the XML Schema in a WSDL document using XML Information Set can be obtained as follows:

- NT is the set of non-terminal symbols. For each element declaration e in the XML Schema there is one $NT_e \in NT$, where e is a QName.
- Σ is the set of terminal symbols. For each element declaration e in the XML Schema there is one $ELEMENT_e \in \Sigma$, where $ELEMENT_e$ is an *Element Information Item* and e is a QName. The arity of $ELEMENT_e$ depends on its declaration in the XML Schema. Additionally there is a symbol $LITERAL \in \Sigma$ with arity 0, that is a *Character Information Item*.
- S is the initial symbol. It is different for each message.
- R is the set of production rules defined with XML Schema following:
  1. $NT_e \longrightarrow ELEMENT_e\ (NT^*)$, if the type of element e is complex.
  2. $NT_e \longrightarrow ELEMENT_e\ (LITERAL)$, if the type of element e is simple.

  RTG grammars derived from the XML Schema in a WSDL description that follows the best practices for interoperability are guaranteed to have, for each non-terminal, exactly one production rule with that non-terminal in the head.

4.2 Relation between Ontologies and RDF data model

Semantic Web Services frameworks operate with semantic data defined by ontologies. Nowadays the logical formalism of choice to build web ontologies is Description Logics. Web languages like OWL-DL and WSML-DL are used to capture domain knowledge used in the Semantic Web Services frameworks. A DL ontology $\mathcal{O}$ are comprised of two major packages:

- The $\mathcal{T}$-Box defines the domain vocabulary and the axioms of the ontology.
- The $\mathcal{A}$-Box defines the individuals of the domain.

As it will be explained afterwards, the terms of ontology vocabulary will be used in the mapping rules between the semantic and the syntactic description of a web service. On the other hand, the $\mathcal{A}$-Box contains the ontological data of the requests ($\mathcal{D}_{rq}$) and responses ($\mathcal{D}_{rs}$) of a web service. The statements of the $\mathcal{A}$-Box can be expressed in the RDF model. This is the information that will be transformed from and to XML.

5. Method to transform trees into graphs: use case for Grounding

As defined in previous sections, data grounding is the process of transforming data from the syntactic level to the semantic representation and back. In order to bridge the gap between syntactic and semantic levels, we need some kind of information that describes how the semantic data can be represented in XML and how the XML data returned from the service can be interpreted using its semantic description.

Our approach for data grounding is based on a ***direct mapping language***. Data structures of WSDL documents are directly mapped to entities of the ontology, in contrast to other approaches such as [28] where intermediate entities are introduced to write the mapping rules. Our mapping rules are clear instructions to perform transformations between XML trees and semantic data graphs. In our approach, we do not need the logic formalism of $\mathcal{O}$ to implement the grounding process, it is merely a structural process that executes a set of mapping rules between non-terminals symbols in a grammar (elements in XML) and the structure of objects descriptions.

Figure 3 summarizes the relations between semantic and syntactic features of a web service. At design time, mapping rules between the RTG grammar (extracted from XML Schema) and the domain ontology are created to link the syntactic and semantic descriptions. At run time, these mapping rules are applied in order to perform lowering and lifting operations. To support the transformation at run time, only the mapping rules and the RTG derived from the WSDL description is required. In other words, the semantic description of the web service and the $\mathcal{T}$-Box of the domain ontology are not used during the actual transformation.

5.1 Mapping language

We propose a mapping language to realize data grounding. Our approach is declarative in the sense that we focus in what data must be transformed and not how to transform it.

Let $\mathcal{G} = (NT, \Sigma, S, R)$ be a regular tree grammar (RTG), extracted from the syntactic description of a web service (WSDL). Let $\mathcal{O}$ be a domain ontology used in the semantic description of the same web service. A tuple $m = (ctx, e, \alpha)$ is a mapping for grounding, where $e \in NT$, $ctx \in NT^*$ (context of e) and $\alpha \in IRI^+$. The mapping is the glue between the semantic and syntactic description. Non-terminal symbols of the grammar are mapped to paths in the graph.

The context of a non-terminal symbol n is the derivation tree from the initial symbol S to n according to the production rules in the grammar. As Sect. 4.1 indicates, there is a bijective correspondence between production rules and non-terminal symbols. Therefore, the sequence of production rules that define the context of a non-terminal symbols can be represented using a sequence of the non-terminal symbols from the initial symbol S. As we have aforementioned (see Sect. 4) our non-terminal symbols are named after the XML elements defined in the XML Schema and those are identified by QNames, therefore we can identify the non-terminal symbols using QNames. Consequently, the context ctx cab be described by a sequence of QNames.

The language of mapping rules for the grounding is defined by the following expression:

$$\mathcal{M} : QName^* \times QName \times IRI^+$$

In the next section, we describe the transformation operations and how they are realized by the functions.

5.2 Transformation operations

There are two main scenarios when data grounding runs:

1. Lowering: the mapping rules guide the process through the semantic model to extract the parameters of the precondition that are used to create the SOAP body of the request. The mapping language supports several entities of the semantic model being mapped to the same non-terminal symbol. Lowering is restrictive with the generated output in order to ensure the creation of valid and well formed XML documents.
2. Lifting: the mapping rules guide the process of parsing the SOAP content and building a set of instances according to postconditions (described using a semantic model) of the web service.

5.3 $\mathcal{L}_{owering}$ function

Given a regular tree grammar, G, a semantic request $\mathcal{D}_{rq}$, and a set of mappings $\mathcal{M}$, we define the lowering process as the transformation function of the semantic information, starting in the node v, to a syntactic representation $T(\Sigma)$

$$\mathcal{L}_{owering} : \mathcal{D}_{rq} \times G \times \mathcal{M} \times v \longrightarrow T(\Sigma)$$

The lowering process starts from the initial symbol S and follows the production rules of the grammar G to create a valid XML Document $T(\Sigma)$. It is driven by the mappings $\mathcal{M}$ that decide how production rules must to be applied. Starting from node v, mappings determine movements in the graph $\mathcal{D}_{rq}$.

Depending on the interpretation of a mapping, the production rule may be applied several times to produce repeated structure. A mapping can be ambiguous. If a non-terminal symbol has been mapped to different paths and they are simultaneously possible in the graph, the mapping can not be resolved and an error is raised.

5.4 $\mathcal{L}_{ifting}$ function

Given a regular tree grammar, G, a syntactic response comprised by a tree of terminals $T(\Sigma)$ and a set of mappings $\mathcal{M}$, we define the lifting process as the transformation function of the syntactic information to a semantic representation $\mathcal{D}_{rp}$:

$$\mathcal{L}_{ifting} : T(\Sigma) \times G \times \mathcal{M} \longrightarrow \mathcal{D}_{rp}$$

The lifting process is realized by bottom-up processing of $T(\Sigma)$. For each terminal symbol its mapping is located and a

subgraph is created by instantiation of the mapping rule. The subgraphs are merged recursively.

Similarly to the lowering process, the lifting process may run into ambiguous cases. If a non-terminal symbol has several mappings in a certain context then it is not possible to determine which subgraph must be instantiated, and consequently an error is raised.

6. Experimentation: the PRAVIA project

E2000 Nuevas Tecnologías[2] is the leading company in the insurance broker sector in Spain. The PRAVIA project was launched by E2000 in partnership with Fundación CTIC with the goal of testing the applicability of semantic web services to this business domain.

Everyday work of the insurance sector and the insurance broker sector includes a huge set of administrative processes for commercial activities, documentation, invoice management, policies, claims... Nowadays, insurance companies expose some digital services to automate some of these processes. Each company defines their own operations and data structures that are interchanged in the transactions of these services. As a result, different companies publish services with different interfaces but with the same purpose.

All the dealers in the insurance sector need to efficiently interchange information with as many insurance companies as possible to perform business operations, such as looking for the best offer for a new customer. A lot of time is lost when the dealers need to enter repeatedly the same information to make similar requests to different insurance companies. Time is also lost to manually study and compare the responses, each one in a different, incompatible format.

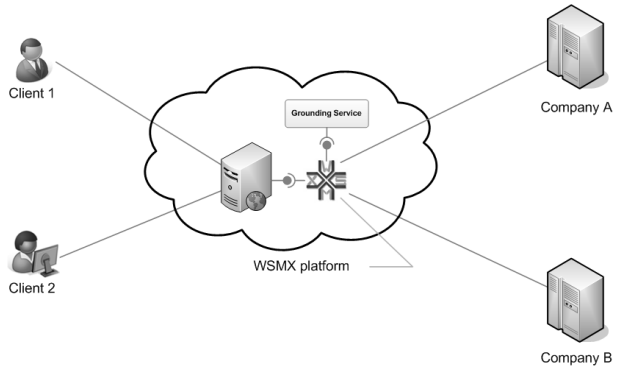

Figure 4. PRAVIA project architecture.

Therefore, it is necessary to use a service-oriented architecture to create a bridge among the dealers and the companies. Such bridge creates a unique access point for communication with all the companies. Development, evolution and maintenance of the bindings for such bridge requires ad-hoc work for each company. As a consequence, they lack the flexibility required to quickly integrate new services, and they create an entry barrier for interface adaptation. The work of the bridge mainly deals with adapting data structures using manual techniques.

In this context, the grounding process and the use of a semantic model (ontology) improve maintainability and scalability. Using this paradigm, all the applications use the same model, and the grounding process automatically adapts the model for each particular web service. When a new company joins the bridge, the only task to be done is to semantically describe its web services according to the semantic model. In a similar fashion, if the operations or services change, the higher abstraction level makes it easier to adapt the system.

For the sake of simplicity, only two business processes and two service providers were chosen. This sample has proven enough to create a realistic environment with non-trivial web services (see Fig. 4). The WSMX platform, the WSMO ontology and the WSML language were selected to deploy the semantic web services technology in this controlled environment. The grounding process provided by the WSMX Communication Manager was found to be in an early stage of development, and it could not address the requirements of the business environment. Therefore, it was replaced with an extended component with the same interface. The new component uses the mapping language and implements the transformation functions described in this paper, and it is one of the outcomes of the PRAVIA project. This experiment has shown the feasibility of our approach for data grounding to fulfill the requirements of a business scenario.

6.1 Annotations in XML Schemas of WSDL Documents

Our approach to data grounding in semantic web services takes into account just the data structure definition (using XML Schema) of the whole WSDL description of a web service. In order to introduce the mapping rules in the XML Schema, we use the elements `xsd:annnotation` and `xsd:appinfo` and its attribute `source`.

Each annotation is interpreted as one or more mapping rules. The non-terminal symbol of the mapping rule is the symbol labeled after the `xsd:element`, see Sect. 4.1. The context is obtained as the list of non-terminal symbols between the current `xsd:element` declaration and the opening declaration of an XML Schema type. The content of an annotation in the `source` attribute is a graph path, described by a list of IRIs separated by whitespaces. By using `xsd:appinfo`, it is possible to introduce multiple mappings associated to the same element.

6.2 Annotations using SAWSDL

Our approach to introduce mappings as annotations in the WSDL Document is similar to the SAWSDL one, although SAWSDL also addresses other entities from the WSDL description, such as types, operations and interfaces.

We initially discarded SAWSDL and decided to use the annotation feature described above due to: 1. temporary lack of support for SAWSDL in the tools, and 2. difficulties to create multiple mappings for the same element. The former is no longer a problem because SAWSDL is now stable and gaining support from tool and framework vendors. Until SAWSDL became a W3C Recommendation, using XML Schema anno-

[2] `http://www.e2000.es`

Table 1. Comparative study of data grounding approaches.

Feature	*Our approach*	*SAWSDL*	*Ad-hoc Implementations (Java, XSPARQL, etc.*
Use of XML Schema annotations	yes, xsd:annotations	no, new attributes	no
Programming skills required	no	yes	yes
Complete solution	yes	Runtime semantics undefined	yes
Usable by domain experts	yes	yes	no
Easy Maintenance and Evolution (changes in ontologies or services)	yes	yes	no
Support for REST Web Services	no	no	yes
Able to perform data mediation	no	yes	yes
Dependence on modeling language	yes	yes	yes

tations was the only standards-compliant approach to decorate a WSDL description with semantic annotations.

Regarding the second issue, it is not obvious how the rules from our mapping language could be integrated in WSDL descriptions using the SAWSDL attributes. The data model of the SAWSDL attributes allows just a list of IRIs (graph path) for each element, and we would like to associate multiple mapping rules to the same element, each one with its own graph path.

7. Comparative study

A qualitative analysis of different approaches to data grounding is shown in Table 1. Firstly, our approach provides a complete solution for data grounding using a full declarative method in a WSDL+SOAP environment. It does not require programming skills, thus improving the maintenance and evolution of the system when changes occur. It also reuses the annotation elements provided by XML Schema to introduce mappings. On the contrary, SAWSDL proposes a partial solution based on a set of WSDL extensions to annotate WSDL Documents with references to semantic entities. There is not any defined semantics to interpret SAWSDL annotations. Both, our solution and SAWSDL, allow business experts to write the mapping rules without a deep knowledge of technology. Finally, all of the revised approaches depend on the semantic modeling language used to build the knowledge base. We remark that our solution lacks support for data mediation while ad-hoc implementations can perform data grounding and mediation in a single step.

8. Conclusions and future work

Our proposal for data grounding relies on a mapping language that describes relations as mapping rules. We pursue a domain-independent and systematic solution that reduces to the minimum the human intervention. These are some remarks about our proposal for data grounding:

- It requires human intervention on design time. The difference falls in which tasks are required and who can do them. Our solution only needs some simple work to map syntactic descriptions with semantic entities. This operation can be carried out by non-technical people, probably using a simple graphical user interface. More specifically, it does not require programming skills.
- It depends on the web service and semantic technologies. We only support WSDL (document based) 1.1 and SOAP 1.x. On the other hand, it is also dependent from the ontology language. We support the current web standards for knowledge representations systems.
- It positively contributes to the maintainability, reliability, robustness, time to repair and cost of the SOA architectures, as a consequence of removing the need of *ad-hoc* developments. This is a distinctive feature of our approach with respect to manually implementing data grounding using languages such as XSLT, XSPARQL or Java.
- It must not be confused with data mediation. Although some shared transformation patterns can be identified, data mediation and data grounding are conceptually different, and their responsibilities should be clearly separated in a successful semantic web services platform.
- It is not a complete solution for semantic web services grounding. Data grounding is able to build requests and to process responses, but that is just a part of SWS grounding. There are other issues that must be addressed separately (e.g., authentication).
- It is a viable solution for data grounding of semantic web services. Without data grounding, SWS platforms can not invoke real web services. However, we can not assure that all SOAP-based web services will be compatible with our solution because heterogeneities are always present due to tools, extensions in specifications, etc.

We reused the experience of previous work (SAWSDL, WSDL-S or OWL-S) in the semantic web services area. Our solution was embedded into WSMX to put it under test in a real environment (non-trivial web services that implement business processes). The results confirm that it is possible to specify data grounding of web services using a mapping language. Anyway, real web services are not limited to those accepting SOAP messages and described by WSDL. Therefore, we are working to extend our solution to support other kinds of service (message protocol and description format), such as REST services, WSDL 2.0, etc. We also study the alignment of our solution with other proposals and recommendations from W3C and OASIS.

In this paper, we have only focused in the description of a mapping language for data grounding. Details of the algorithms of the lifting and lowering functions together with a complete example and execution trace will be the subject of a subsequent publication.

References

[1] R Akkiraju, J Farrell, J Miller, M Nagarajan and M T S et al., Web Service Semantics-WSDL-S, W3C member submission, W3C, 2005, http://www.w3.org/Submission/WSDL-S/.

[2] D Anicic, M Brodie, J D Bruijn, D Fensel, S Heymans, J Hoffmann, M Kerrigan, J Kopecky, R Krummenacher, H Lausen, A Mocan, I Toma and M Zaremba, Semantically enabled service oriented architectures: A manifesto and a paradigm shift in computer science, Tech. rep., In Proceedings of WICI International Workshop on Web Intelligence (WI) meets Brain Informatics (BI) (WImBI 2006), 2006.

[3] S Battle, A Bernstein, et al., Semantic Web Services Framework (SWSF), W3C member submission, W3C, 2005, http://www.w3.org/Submission/SWSF/.

[4] H Bohring and S Auer, Mapping XML to OWL Ontologies.. Leipziger Informatik-Tage 2005, K P Jantke, K P Fähnrich, and W S Wittig, Eds., Vol. 72 of LNI, GI, pp. 147–156.

[5] D Booth, H Haas, F McCabe, E N (until October 2003), M C (until March 2003), C F (until March 2003) and D O (until March 2003), Web Services Architecture, W3C working group note, W3C, 2004, http://www.w3.org/TR/ws-arch#introduction.

[6] J d Bruijn, M Ehrig, C Feier, F Martín-Recuerda, F Scharffe and M Weiten, Ontology mediation, merging and aligning, In Semantic Web Technologies. Wiley, 2006.

[7] M H Burstein, C Bussler, M Zaremba, T W Finin, M N Huhns, M Paolucci, A P Sheth and S K Williams, A Semantic Web Services Architecture.. IEEE Internet Computing, Vol. 9, No. 5, 2005, pp. 72–81.

[8] L Cabral, J Domingue, E Motta, T R Payne and F Hakimpour, Approaches to Semantic Web Services: an Overview and Comparisons. ESWS 2004, pp. 225–239.

[9] J Calladine and P Downey, Xml Schema and Web Services. W3C Workshops on XML Schema 1.0 User Experiences 2005.

[10] H Comon, M Dauchet, R Gilleron, C Löding, F Jacquemard, D Lugiez, S Tison and M Tommasi, Tree Automata Techniques and Applications, 2007, release October, 12th 2007.

[11] A Corradini, U Montanari, F Rossi, H Ehrig, R Heckel and M Loew, Algebraic Approaches to Graph Transformation, part I: Basic Concepts and Double Pushout Approach, Tr-96-17, Università di Pisa, Dipartimento di Informatica, 1996.

[12] J Cowan and R Tobin, XML Information Set (Second Edition), W3C recommendation, W3C, 2004, http://www.w3.org/TR/xml-infoset/.

[13] J de Bruijn, C Bussler, J Domingue, D Fensel, et al., Web Service Modeling Ontology (WSMO), W3C member submission, W3C, 2005, http://www.w3.org/Submission/WSMO/.

[14] D C Fallside and P Walmsley, XML Schema Part 0: Primer Second Edition, W3C recommendation, W3C, 2004, http://www.w3.org/TR/xmlschema-0/.

[15] J Farrell and H Lausen, Semantic Annotations for WSDL and XML Schema, W3C recommendation, W3C, 2007, http://www.w3.org/TR/sawsdl/.

[16] G Klyne and J J Carroll, Resource Description Framework (RDF): Concepts and Abstract Syntax, W3C recommendation, W3C, 2004, http://www.w3.org/TR/rdf-concepts/.

[17] J Kopecký, Simple RDF to XML Data Grounding (slides), 2005.

[18] J Kopecký, M Moran, T Vitvar, D Roman and A Mocan, Eds., D24.2v0.1. WSMO Grounding 2007, WSMO.

[19] J Kopecký, D Roman, M Moran and D Fensel, Semantic Web Services Grounding.. AICT/ICIW 2006, IEEE Computer Society, p. 127.

[20] J Kopecký and A Schütz, D1.2.1 WSMO grounding in SAWSDL, SOA4All deliverable d1.2.1, European Project-FP7, 2008, http://www.soa4all.eu.

[21] J Kopecký, T Vitvar, C Bournez and J Farrell, SAWSDL: Semantic Annotations for WSDL and XML Schema.. IEEE Internet Computing, Vol. 11, No. 6, 2007, pp. 60–67.

[22] I Marinchev and G Agre, Semantically Annotating Web Services Using WSMO Technologies. CYBERNETICS AND INFORMATION TECHNOLOGIES, Vol. 5, 2005, p. 12.

[23] D Martin, M Burstein, et al., OWL-S: Semantic Markup for Web Services, W3C member submission, W3C, 2004, http://www.w3.org/Submission/OWL-S/.

[24] A Mocan and E Cimpian, Eds., D13.3v0.2 WSMX Data Mediation 2005, WSMO.

[25] A Mocan and E Cimpian, WSMX Data Mediation, WSMX working draft, Deri, 2005, http://www.wsmo.org/TR/d13/d13.3/.

[26] M Murata, D Lee and M Mani, Taxonomy of XML Schema Languages using Formal Language Theory. Extreme Markup Languages Montreal, Canada, 2001.

[27] F Noy and A Musen, Evaluating Ontology-Mapping Tools: Requirements and Experience, 2002.

[28] K Pantschenko, O Noppens and T Liebig, Grounding Web Services Semantically: Why and How?. W3C Workshop on Frameworks for Semantics in Web Services 2005.

[29] D Roman, J de Brujin, A Mocan, I Toma, H Lausen, J Kopecký, C Bussler, D Fensel, J Domingue, S Galizia and L Cabral, Semantic Web Technologies. Trends and research in ontology-based systems, Wiley, 2006.