Semantic and Syntactic Data Flow in Web Service Composition

Freddy Lécuë\textsuperscript{1}, Samir Salibi\textsuperscript{1}, Philippe Bron\textsuperscript{1}
\textsuperscript{1}France Telecom R&D, France
4 rue du clos courtel, F-35512 Cesson Sévigné
\{(firstname.lastname)@orange-ftgroup.com\}

Aurélien Moreau\textsuperscript{2}
\textsuperscript{2}Laboratoire d’Informatique de Paris 6 (LIP6)
104 avenue du Président Kennedy
75016 Paris, France
\{(aurelien.moreau)@lip6.fr\}

Abstract

Automation of Web service composition is one of the most interesting challenges facing the Service Oriented Computing today. From this challenge, many issues such as control flow, data flow, verification, execution monitoring, or recovery actions (e.g., compensation) follow. In this paper, we focus on automated data flow in Web service composition. The semantic Web, as an evolving extension of the current Web, seems a key initiative to overcome the latter issue. However, even if some approaches focus on discovering potential semantic connections between Web services, few or none of these tackle implementations issues related to XML messages management at syntactic level. In this direction, we present an approach for performing automated data flow in Web service composition by i) exploiting semantic matchmaking between Web service parameters (i.e., outputs and inputs) to enable their connection and interaction, and ii) adapting XML database solutions, specifically XML Schema mapping, to perform syntactic data transformation and integration of exchanged messages. Our system is implemented and interacting with Web services dedicated on a Telecom scenario. The preliminary evaluation results showed not only high efficiency and effectiveness of the proposed approach but also complementarity of the semantic matchmaking and syntactic mapping to achieving data flow in Web service composition.

1 Introduction

The semantic web, i.e., the web of meaning is considered as the new vision and extension of the current web that try to give semantic to the web resources [4]. Web services in the semantic web are enhanced using rich description languages (through Description Logics [3]) such as the Web Ontology Language (OWL) [27]. In this way Semantic web services [28] are web services that have been enhanced with formal semantic descriptions where OWL-S [1], WSMO [11] or SA-WSDL [26] may be used to describe them. Semantic Web services can, in turn, use these descriptions to reason about web services and automate their use to accomplish goals specified by end-users including intelligent and automated discovery, selection and composition.

Web services proliferation over the web implies difficulties to find specific services that can perform specialized tasks. Nevertheless, a composition of some existing services is an alternative and promising approach to overcome this problem by reusing existing services. Starting from an initial set of services, Web service composition aims at selecting and inter-connecting Web services provided by different partners according to a goal to achieve.

Towards the latter challenge, many issues such as control flow, data flow, verification, execution monitoring, or recovery actions (e.g., compensation) have followed. In this work, we focus on data flow in semantic Web service composition i.e., data integration in composition through its input and output parameters. Even if some approaches focused on discovering potential semantic connections between Web services (e.g., [14]), few or no solution suggested a valid and executable program that has a correct data flow. Indeed, once one retrieves composable services through a semantic matchmaking, such a feature is still missing. Thus, the compositions that come by semantic matchmaking are meaningful, but rarely executable. One of the underlying issues consists in (heterogeneous) data integration between (heterogeneous) Web services involved in a composition. Towards this issue, we present an approach for performing automated data flow in Web service composition by i) exploiting matchmaking between service parameters (i.e., outputs and inputs) to guarantee their semantic connection and interaction, and ii) providing the syntactic data mappings (specifically XML Schema mapping) necessary to integrate exchanged messages of heterogeneous services in an appropriate way [25]. Roughly speaking, we suggest to adapt XML database integration techniques by including some semantic Web Service processing to generate an executable transformation in a standard Web technol-
ology, namely XSL, to support data integration.

The remainder of this paper is organised as follows. In the next section we present the scenario wherein we integrated the suggested approach. Section 3 briefly reviews i) causal links i.e., a semantic link between Web services and ii) requirements on syntax mapping. Section 4 presents the semantic and syntax based approach to perform data flow in Web service composition. Section 5 presents the prototype and some experiment results regarding the syntactic and semantic valuation of causal links. Section 6 briefly comments on related work. Finally section 7 draws some conclusions and talks about possible future directions.

2 A Motivating Use Case

The suggested approach is being exploited in a R&D Project, conceived by France Telecom. In this project, we focused on an online service composition tool. The aim of this tool is to enable end-users to create composite services on the fly, without developing code. Since interactions between Web services at run time is still an open issue, we invest on a automated method to perform data flow in a composition. A component integrating semantic and syntactic data flow in composition has been required. In this direction we propose a tool that support creator of composite services to perform data mapping. The ultimate goal is to ensure dynamic generation of data mapping between parameters of services, in regard to their semantic connections. Figure 1 shows a portion of the ACN Ontology1 (105 concepts and 37 properties) used to describe the domain i.e., concepts of the ontology refer to all input and output parameters of services. Such a conceptualization enables first to model semantic connections between service parameters.

Suppose an interaction of two Web services. First of all we perform semantic matchmaking between parameters (as concepts) of those services by means of a domain ontology and its reasoner. This gives matchmaking between parameters with scores. A syntactic analysis is then performed. The latter analysis is based i) on Xsd structure and ii) on the semantic report, provided by the first phase.

In the rest of the paper, we consider two Web services i.e., a subset of the 29 services included in the real scenario:

- an AddressBook service, that, manages users contacts. It then returns a list of contacts defined by its LastName, FirstName, Address (e.g., Country, City, CountCode, ZipCode).
- an Itinerary service starting from two Addresses i.e., CountryLabel, CountryCode, City, Street, ZipCode, calculates the Path i.e., any door-to-door route and supplies detailed driving Directions.

\[ \text{Address} \equiv 1\text{City} \sqsupseteq 1\text{Country} \sqsupseteq 1\text{ZipCode} \]
\[ \text{Country} \sqsubseteq \text{CountryName} \]
\[ \text{CountryLabel} \sqsubseteq \text{CountryName} \]
\[ \text{CountCode} \equiv \text{CountryCode} \]
\[ \text{ZipCode} \sqsubseteq \text{Lastname} \sqsubseteq \text{T}, \text{FirstName} \sqsubseteq \text{T}, \text{City} \sqsubseteq \text{T} \]

**Figure 1. Sample of an ACN Ontology T.**

At run time, the suggested tools aims at suggesting relevant data mapping in an automated way. From the generated data mapping, end-user can then validate or modify the suggested data mapping. Finally the mapping rules are generated in xsl format.

3 Background

First of all, we present causal links and finally we briefly describe data integration in the Web.

3.1 Composition & its Causal Links

In the semantic web, parameters (i.e., input and output) of web services refer to concepts in an ontology \( T \) (e.g., Figure 1), where the OWL-S profile [1], WSMO capability [11] or SA-WSDL [26] can be used to describe them. At functional level, web service composition consists in retrieving some semantic links between output parameters \( Out_s_i \in T \) of services \( s_i \) and input parameters \( In_s_j \in T \) of other services \( s_j \). Such a link i.e., causal link [15] \( cl_{i,j} \) (Figure 2) between two functional parameters of \( s_i \) and \( s_j \) is formalized as \( \langle s_i, Sim_T(Out_s_i, In_s_j), s_j \rangle \). Thereby \( s_i \) and \( s_j \) are partially linked according to a matching function \( Sim_T \). This function expresses which matching type is employed to chain services. The range of \( Sim_T \) is reduced to the four well known matching type introduced by [23] with the extra type Intersection [17]:

- **Exact** If the output parameter \( Out_s_i \) of \( s_i \) and the input parameter \( In_s_j \) of \( s_j \) are equivalent; formally, \( T \models Out_s_i \equiv In_s_j \).
- **PlugIn** If \( Out_s_i \) is sub-concept of \( In_s_j \); formally, \( T \models Out_s_i \sqsubseteq In_s_j \).
- **Subsume** If \( Out_s_i \) is super-concept of \( In_s_j \); formally, \( T \models In_s_j \sqsubseteq Out_s_i \).
- **Intersection** If the intersection of \( Out_s_i \) and \( In_s_j \) is satisfiable; formally, \( T \models Out_s_i \cap In_s_j \sqsubseteq \bot \).
- **Disjoint** Otherwise \( Out_s_i \) and \( In_s_j \) are incompatible i.e., \( T \models Out_s_i \cap In_s_j \sqsubseteq \bot \).

\[^1\text{Distributed ontologies are not considered here but are largely independent of the problem addressed in this work.}\]
Example 1. (Matchmaking and Functional Parameters)
Suppose AddressBook and Itinerary be the two services of the motivating
scenario. It is obvious that Country i.e., the output parameter of AddressBook
semantically matches CountryLabel i.e., an input of Itinerary with an Intersection
match type.

Example 2. (Semantic Causal Links)
Suppose the two semantic Web services of the motivating
scenario. The four causal links $c_{i,j} \leq 4$ follow:
- $(AddressBook, Sim_{T}(City, City), Itinerary)$;
- $(AddressBook, Sim_{T}(CountCode, CountryCode), Itinerary)$;
- $(AddressBook, Sim_{T}(ZipCode, ZipCode), Itinerary)$;
- $(AddressBook, Sim_{T}(Country, CountryLabel), Itinerary)$.

The latter matching function enables, at design time, finding some levels of semantic
(in)compatibilities among independently defined web service descriptions.

Since a composition of Web services consists of a partial
order of web services wherein these services are semanti-
cally chained by causal links, web service composition can be
studied as a causal links composition.

3.2 Data integration in the Web

In our approach, the XSL technology is chosen as XML
transformation language to ensure integration of messages exchanged by Web services. The XSL technology is sup-
ported by most frameworks, stands as a widely spread standard and comes with several tools. The XSL language is
declarative: template rules define how to handle a node matching a particular XPath-like pattern (MATCH), if the
processor should happen to encounter one in the (MODE) state. The contents (CONTENT) consists of a tree of operation
which builds the target document. A basic XSL template is defined as follows.

```xml
<xsl:template match=' {MATCH} ' mode=' {MODE} '>
  <xsl:element name='CONTENT'/>
</xsl:template>
```

The result tree is built by combining some expressions as:
- `<xsl:element name=' {ELEMENT} '/>` to create a new node (ELEMENT) in result tree,
- `<xsl:apply-templates select=' {SELECT} ' mode=' {MODE} '/>` to apply a new template on the (SELECT) source in the state (MODE),
- `<xsl:value-of select=' {SELECT} '/>` to copy the value of (SELECT) source in current target node.

4 Semantic and Syntactic Mapping

In this section we assume that the control flow (i.e.,
the ordering of services) of compositions is computed in a
pre-processing step. Many approaches such as [19, 7]
can achieve this goal in an automated way. Moreover, all
functional parameters of Web services are supposed to be
known at i) syntactic level i.e., defined by XML artefacts
through WSDL and ii) semantic level i.e., annotated by De-
scription Logsics (DL) concepts through SA-WSDL annota-
tions in XML Schemas.

From these we perform two main tasks i.e., i) semantic
matchmaking between output and input parameters and ii)
syntactic mapping between their XML Schemas.

4.1 Semantic Matchmaking

Semantics provides a high-level reasoning for automatic
discovery of connections between Web services and data. The semantic matchmaker aims at retrieving relevant
matching between output and input parameters of Web services as Description Logsics (DL) concepts in a
domain ontology. The standard reasoning inferences Exact (\( w(Exact) \) i.e., the numerical value assigned to Exact
is 1), PlugIn (\( w(PlugIn) \) is \( \frac{3}{4} \)), Subsume (\( w(Subsume) \) is \( \frac{1}{4} \)), Intersection (\( w(Intersection) \) is \( \frac{1}{4} \)) and Disjoint
(\( w(Disjoint) \) is 0) are used to semantically value the qual-
ity of the latter matching. It is then straightforward to dis-

\( c_{i,j} \)

\( c_{i,j} \)

\( c_{i,j} \)

\( c_{i,j} \)

\( c_{i,j} \)

\( c_{i,j} \)

\( c_{i,j} \)

\( c_{i,j} \)

At the beginning of the semantic matchmaking interp
The result tree is built by combining some expressions as:

- `<xsl:element name=' {ELEMENT} '/>` to create a new node (ELEMENT) in result tree,
- `<xsl:apply-templates select=' {SELECT} ' mode=' {MODE} '/>` to apply a new template on the (SELECT) source in the state (MODE),
- `<xsl:value-of select=' {SELECT} '/>` to copy the value of (SELECT) source in current target node.

4. Semantic and Syntactic Mapping

In this section we assume that the control flow (i.e.,
the ordering of services) of compositions is computed in a
pre-processing step. Many approaches such as [19, 7]
can achieve this goal in an automated way. Moreover, all
functional parameters of Web services are supposed to be
known at i) syntactic level i.e., defined by XML artefacts
through WSDL and ii) semantic level i.e., annotated by De-
scription Logsics (DL) concepts through SA-WSDL annota-
tions in XML Schemas.

From these we perform two main tasks i.e., i) semantic
matchmaking between output and input parameters and ii)
syntactic mapping between their XML Schemas.

4.1 Semantic Matchmaking

Semantics provides a high-level reasoning for automatic
discovery of connections between Web services and data. The semantic matchmaker aims at retrieving relevant
matching between output and input parameters of Web services as Description Logsics (DL) concepts in a
domain ontology. The standard reasoning inferences Exact (\( w(Exact) \) i.e., the numerical value assigned to Exact
is 1), PlugIn (\( w(PlugIn) \) is \( \frac{3}{4} \)), Subsume (\( w(Subsume) \) is \( \frac{1}{4} \)), Intersection (\( w(Intersection) \) is \( \frac{1}{4} \)) and Disjoint
(\( w(Disjoint) \) is 0) are used to semantically value the qual-
ity of the latter matching. It is then straightforward to dis-
hing causal links valued by different matching functions.

Since the set of relevant Web services is supposed to be
known, the main step of the semantic matchmaker consists in
computing all matching values between output and input parameters of Web services. From this step, we obtain a
set of triple (\( Out_{s_i}, In_{s_j}, Sim_{T}(Out_{s_i}, In_{s_j}) \)) where
\( Out_{s_i} \) is an output parameter of service \( s_i \), \( In_{s_j} \) is an input
parameter of service \( s_j \) is an output parameter of a service
and \( Sim_{T}(Out_{s_i}, In_{s_j}) \) a value in \( \{0, \frac{1}{4}, \frac{3}{4}, \frac{1}{4}, 1\} \).

Finally the set of triple with a strict positive value are pro-
duced to the syntactic engine through an input format named
“ConceptMapping”, “ConceptMapping” and XML Schema
together with SA-WSDL annotations will be the two inputs
of the syntactic engine.
The computational complexity of this step is square in the cardinal of input and output parameters of services. In case the ontology is not too complex ($\mathcal{ALN}$ seems a good trade-off between expressivity and complexity) and with a correct number of concepts, the semantic reasoning is less than one millisecond for standard reasoning.

Most of semantic engines (and specifically the one we use) deal with semantic reasoning of DL descriptions of functional parameters. Unfortunately the latter engines do not provide any features for data transformation, or only at DL level. Indeed DL concepts are not XML message types and the structured nature of XML is not captured in such concepts. Even if SA-WSDL enables annotation of XML types with semantic concepts and entice the developer to define respectively data-to-ontology and ontology-to-data transformations with the liftingSchemaMapping and loweringSchemaMapping extensions, this standard does not provide any guideline about these two extensions. Concept mapping and transformation generations are out of the scope of SA-WSDL, thus automatic generation of transformations are not explored yet. Here, these two extensions are not considered since they are rarely available.

4.2 Data Adapter Generation

At syntactic level, Web service standard such as WSDL defines the input and the output of a service in term of messages. In most of case a semantic parameter is a part of a message constrained by a XML Schema document. The relation between the semantic and the syntactic layer is simplified in Figure 3.

![Figure 3. Semantic and Syntactic Relation.](image)

Once a Web service composition with its semantic matchmaking are retrieved, the semantic engine gives way to the syntactic engine. Even if causal links are appropriate in a semantic context they do not guarantee syntactic compatibility between Web services. Therefore the syntactic engine aims not only at reusing the semantic links but also at producing an XSL transformation as adapter.

First of all, we apply a procedure of structural analysis. In such a step, syntactic links between source and target elements of XML Schema are retrieved. From these pairs, syntactic scores are computed. The set of resulting links is then stored in a document called XSDMapping. This document is very valued added, since it can be modified and validated by, for instance, an expert or another process. Finally, an XSL document is generated from these links (procedure designated as XSL generation). In the following, we detail the structural analysis procedure and the XSL generation.

4.2.1 Structural Analysis

The structural analysis process aims at discovering the structural relationships between input and output XML Schemas. It capitalizes i) on the semantic results produced by the semantic engine and ii) on syntactic and structural indications available in XML Schemas. From this, no transformation is generated, but several links are returned, especially to ensure modification and validation. These links are then exchanged with both the user interaction component and the transformation generation component, thanks to XSDMapping documents.

The overall process is inspired from Boukottaya’s thesis [5]. This solution has been initially developed for database systems, but we adapt it here to the semantic Web service technology. These adaptations consist briefly in the three following major changes:

- In the original thesis the input values of similarity are provided by a lexical dictionary like WordNet [10]. In our solution the similarity values are import from the semantic matchmaking. Hence, the “linguistic matching” component of Boukottaya’s process becomes a “semantic import” component in our solution.

- The “Node context similarity” component in the original system uses some distances which are inappropriate. An edit distance [16] and a Hausdorff [22] distance are preferred in this work. Indeed these edit distances are more suitable and efficient in our context.

- Finally the whole set of complete transformation operations proposed for mappings rules (implemented in XSDMapping) is not implemented in our approach. In contrary, we perform it by introducing two novel operations.

As shown in Figure 4, two analyses are carried out in parallel. The first analysis is a generic procedure, while the second analysis is an ad-hoc approach to recognize known patterns in the XML Schemas. These two branches are joined by a node similarity search that produces a XSDMapping document. Both analyses use a structural representation of the XML Schemas called SchemaGraph. This structure is a graph view of XML Schema types. Each node corresponds to an XML Schema element. The edges represent
relations between nodes, as the container-contained relation or the element-attribute relation.

In the Semantic Import component, for each pair of source-target nodes in a SchemaGraph, the semantic value computed between the source concept and the target concept is copied into a "similarity matrix". The resulting similarity matrix constitutes the root of the two analysis branches.

In the Designer type hierarchy analysis branch, an ad-hoc study is performed. It aims at discovering some special inheritance configurations. Currently we have not explored this module and we simply generate a "union"-type rule for all nodes with the same complex type (according to XML Schema), under certain conditions. This component is still to be developed.

In the DataType compatibility and Node context similarity branch, the matrix is updated according to compatibility of data types. The data type compatibility modifier is pre-computed from built-in XML schema data types. After what, a new matrix is constructed, considering the context of each node. A node context is divided into three sections: i) the ancestor context, including all parents, ii) the children context containing all the direct children and finally iii) the leaf context consisting of the leaves dominated by the node. For each pair of nodes with a similarity coefficient above a threshold, the distance between the respective contexts is computed. Contrariwise to Boukottaya's proposal, we choose an edit distance for ancestors and ordered direct children and a Hausdorff distance for non-ordered direct children and the leaf context.

Finally, the last component, called Discovery of nodes and edges matches, uses the context similarity matrix and information provided by the designer type hierarchy as input to generate rules with relevance scores that are stored in XSDMapping. We apply exactly Boukottaya’s algorithm to discover node matching.

An XSDMapping document is an implementation of the “mapping rules” of Boukottaya’s thesis. It is composed of a set of links between elements of source and target schemas.

A relevant score is further added for each links, this is required to select the best rule. We choose to implement the “connect”, “union”, “split” and “merge” transformation operations. Moreover, we add two new operations called “xsl” and “constant”. An “xsl” operator allows the embedding of the XSL user code (the code is copied verbatim). And “constant” operator fills attributes or leaves with a user defined constant. These both new types increases the power of user interaction for complex mappings. They are not generated but can be added by the users.

4.2.2 XSL Generation

The XSL generation component aims at generating the adapters. An adapter is an XSL document. It is produced using a XSDMapping and the XML Schemas of provided and requested parameters. The transformation is performed at runtime in order to prepare the input data for the service to be called. The adapter generation process is twofold. First, a XSDMapping rule is chosen for each node of target SchemaGraph. This rule must contain the node within its targets and must be the best according to the relevance score. “N.rule” denotes the rule associated with node N. Second, a depth-first traversal of the target SchemaGraph is performed. For each node, an XSL template is generated and three variables are maintained in a manner analogous to attribute grammars [8]. The two synthesised variables |Select and |Mode represent respectively the relative path and the state to be, to apply the template. The inherited variable |Srcs contains the set of potential current nodes in the source SchemaGraph. This last variable is used to create the relative path.

We present two simplified examples of the XSL patterns in Figure 5. The first box indicates the type of the corresponding rule, where N represents the current node. The second box presents the update rules for the variables. Finally the third box contains the generated template. The connect(c) type is the XSDMapping connect link when the target node is complex (c). A similar pattern is defined for each XSDMapping link type.

Example 3. Service Messages and Adapter

Suppose the input message with the semantic annotations (SA-WSDL) for the Itinerary service looks like

```
<iti_op4>
  <origin sawsdl:modelRef='#Address'>
    ....
  </origin>

  <destination sawsdl:modelRef='#Address'>
    <city sawsdl:modelRef='#City '/>
    <country sawsdl:modelRef='#CountryCode '/>
    <countryLabel sawsdl:modelRef='CountryLabel '/>
    <zipCode sawsdl:modelRef='ZipCode '/>
    <streetLabel sawsdl:modelRef='StreetLabel '/>
  </destination>
</iti_op4>
```
Figure 5. XSL pattern examples

and the AB output looks like

```
<retrieveContactResponse>
  <retrieveContactReturn>
  </retrieveContactReturn>
</retrieveContactResponse>
```

Then, a part of the generated XSL adapter is

```
<xsl:template match='retrieveContactResponse'/>
<xsl:element name='tti_op4'/>
<xsl:element name='destination'/>
  <xsl:apply-templates mode='s1' select='.'/></xsl:element>
  <xsl:apply-templates mode='s2' select='.country'/></xsl:element>
  <xsl:apply-templates mode='s3' select='.countryLabel'/></xsl:element>
</xsl:template>
```

5 Prototype & Experimental Results

5.1 Prototype

Several scenarios have been achieved to prove the feasibility of our semantic and syntactic approach. These scenarios have been performed by means of a domain ontology, especially to infer semantic matchmaking. It contains all the concepts referred in semantic Web services. Figure 6, 7 and 8 illustrates screenshots of our online service composition tools. In a nutshell, the latter figure illustrates mapping generated associated with their scores. First of all Figure 6 illustrates only the semantic matchmaking between two parameters of services.

This picture shows that all parameters of the target Xsd are mapped. The scores of City, CountryCode, StreetLabel and ZipCode are equal to 1 since these parameters have the same concepts as AddressBook ones i.e., City, countCode, AddressLine1, AddressLine2 and ZipCode. The CountryLabel is labeled by a lower score i.e., 0.25, since it does not have exactly the same concept as Country. Indeed there is only a Intersection relation between them.

Let focus on the syntactic mapper in Figure 7. The terms City and ZipCode are matched with a score 0.74 since they have same syntax. Concerning CountryCode and CountryLabel, the syntax is close to the parameters from AddressBook: CountCode and Country. That is why the score is less than 0.7. The main difference with the previous scenario is concerning StreetLabel. The latter term does not have any mapping since its syntax is very different from AdressLine.
Finally we generate both matching. In such a case, all parameters have associated mapping, which is a real benefit for Web service composition. Indeed all parameters are mapped automatically; the developers or end-user is only required to check if the suggested mapping links are valid. In case the ontology is more complex, the suggested component is able to retrieve much more complex data mapping.

5.2 Experimental Results

In the experiments the PC used for running the prototype system had the configuration of Intel(R) Core(TM)2 CPU, 1.86GHz with 512 RAM. The PC runs Linux-gnu (2.6.12-12mdk) and Java 2 Edition v1.5.0_11.

We conducted experiments using the implemented prototype system to evaluate our approach. More specifically our approach has been evaluated on the motivating scenario wherein a step of semantic matchmaking between output and input parameters is required. From this, the syntactic mapping is achieved. Before the data integration and mapping steps, then end-user interacts with the system to approve or refuse the results e.g., the end-user can reject the data mapping between country and countryLabel since their semantic and syntactic similarities are both below the critical threshold of 0.5.

Here, we compare performance results of the semantic and syntactic level of our approach in Figure 9. In a nutshell the semantic and syntactic tasks are tested on causal links $c_{i,1 \leq i \leq 4}$ illustrated in Example 2.

5.3 Performance Evaluation

According to the experimentation results drawn in Figure 9, the semantic matching step is more time consuming than the syntactic mapping step. Indeed the former step requires complex tasks such as ontology loading and classification, DL reasoning on semantic concepts whereas the latter task is limited to fast computation of edit distance and syntactic data transformation.

6 Related Work

Data integration problem is not a new issue. In particular, it may be found in distributive heterogeneous database. A lot of efficient mediation architectures have been developed with an architectural [13] and with a semantic [6] point of view. Many of these solutions are based on a relational model. Only more recently XML mediators have been developed [12].

Schema matching techniques have been explored in many works and particularly in [18, 20], and [25] for a survey. The transformation generation and schema matching tasks are two separate ones but are still connected issues. However, in many case the schema matching approach does not provide an easy solution to generate corresponding transformations. The Clio system [21] provides the integration of XML and relational schemas in common frameworks and allows transformation generation after a schema matching operation. However Clio is a big one and it requires a lot of user interactions not compatible with our requirements.

Data integration in Web Services is poorly covered. Works as [24, 2, 9] explore some solutions. Even if they present an integration architecture for service composition, the semantic potential is not considered for generating the adapters. In the semantic Web service area [14] studied an
approach to perform semantic matchmaking between services. However the syntactic mapping between parameter of services is not addressed in this work. This is a major limitation for automation of Web service composition.

7 Conclusion and Future Work

We have presented an approach for performing automated data flow in Web service composition by i) exploiting semantic matchmaking between service parameters (i.e., outputs and inputs) to enable their connection and interaction, and ii) adapting XML database solutions, specifically XML Schema mapping, to perform syntactic data transformation and integration of exchanged messages. The preliminary evaluation results showed not only high efficiency and effectiveness of the proposed approach but also the complementarity of the semantic matchmaking and syntactic mapping to achieving data flow in Web service composition.

It is straightforward to generalize our work with Web services outside the Telecom domain such as banks, insurance, and so on. Moreover the suggested tool can be used to provide additional meta information required by services/applications integration (through their semantic and syntactic data flow) and related issues in Information systems, B2B, SaaS. In this direction, adding a semantic based layer provides more flexibility for information exchanges between heterogeneous applications.

Since ontologies in Telecom domain are not yet well developed, our approach is limited on a unique, local ontology, restricting interoperability e.g., with other Telecom actors. The un-structured format of WSDL interfaces requires designers to model and specify WSDL in a good shape.

References