Handling semantic conflicts in DaaS composition: A service mediation approach

Idir Amine Amarouche*, Michael Mrissa† and Zaia Alimazighi*
*Université des Sciences et de la Technologie H.Boumediene BP 32 El Alia 16111
Bab-Ezzouar, Alger, Algeria
Email: I.A.Amarouche@gmail.com , alimazighi@wissal.dz
†Université de Lyon 1, CNRS LIRIS-UMR5205 43, bd du 11 novembre 1918, Villeurbanne, F-69622, France
Email: Mrissa.Michael@liris.cnrs.fr

***Abstract*—In the domain of DaaS[1], completing a query means calling many services which are heterogeneous and built independently from the context in which they will be used. This heterogeneity leads to several compatibility problems during DaaS composition. In order to solve them, we propose a semantic description model which allows context characterization. The proposed model enables data mediation in the composition for resolving the conflicts caused by heterogeneities between DaaSs. We rely on two-layered mediated ontology for deriving automatically DaaSs compositions that incorporate necessary mediation services. A preliminary evaluation has been performed based on our initial investigation leading to better improvement.**

I. Introduction

As commonly agreed, Web services fall into two categories depending on their functionality world-altering services and information-providing ones [1]. The latter ones are regarded as specific database views with binding patterns. Thus, the DaaS composition problem is reduced to a query rewriting problem in the data integration field. Doing so, in the context of semantic Web, several works [2], [3], [4], [5] proposed DaaS compositions approaches with the help of query rewriting techniques. The key idea behind these approaches is to describe DaaSs as Parametrized RDF Views (*PRVs*) over mediated ontology to capture their semantics in a declarative way. Defined views are then used to annotate DaaSs description files (e.g. WSDL files) and are exploited to automatically compose DaaSs.

However, in the Internet environment there are several reference ontologies that formalize the same domain knowledge. The domain ontology cannot provide contextual definitions and contextual data structures to represent the diversity of perceptions and focuses. Thus, the construction of a mediated ontology unifying all existing representations of real-world entities in the domain is a strong limitation to interoperability between DaaS. As a result, even though we can automatize DaaS composition by semantically annotating their descriptions, the limitation cited previously raises semantic conflicts between pieces of data exchanged during DaaS composition. In order to overcome this problem, mediation mechanisms based on service-oriented approach to implement mediators must be inserted into DaaS compositions.

[1]DaaS: Data-as-a-Service or information-providing service

Despite the fact that previously cited works adopt a DaaS model similar to *PRVs* [2] over a mediated ontology, they differently cover the life cycle of a DaaS composition ([4], [5] cover only DaaS discovery and selection but [2], [3] cover the whole DaaS composition life cycle) and none of them had considered the data mediation aspect in DaaS composition.

In general, existing composition frameworks do not include the possibility to detect and resolve semantic conflict between data exchanged during DaaS composition. Indeed, solving semantic conflicts (ontological reference, unit,....etc) and performing a meaningful composition have to be achieved by describing the conversion of data between different semantic representations.

In a nutshell, we propose an approach to integrate data provided by several DaaSs using two-layered knowledge representation, based on *Domain Ontology (DO)* and *Contextual Ontology (CO)* for automatically deriving DaaSs compositions with appropriate mediation services to carry out data conversion between interconnected DaaS. Doing so, on the basis of the DaaS model and the query rewriting approach for DaaS composition proposed in [2], our main contributions in the paper are summarized in two points. Firstly, we propose an extension to the RDF-based DaaS model. Specifically, we represent DaaSs and mediation service as an *Extended PRVs* over ontologies (DO and CO). We adopt SPARQL, the de facto query language for the Semantic Web, for posing queries over DaaS services. Secondly, we propose an enhanced query processing approach to automatically detect and resolve semantic conflict in DaaS composition.

In our approach, SPARQL queries specified over a mediated ontology, are reformulated in terms of available DaaSs based on the defined *PRVs*. Thus, since the data provided and required by individual DaaSs may be bound to different semantics, we propose a mechanism that automatically inserts mediation services in order to resolve the semantic incompatibilities detected in the generated DaaS compositions.

The paper is organized as follows: In section II we present a motivation example within service oriented system to introduce the need for mediation service in DaaS composition

[2]Unlike [2] the previously cited works do not consider input-output semantic relationship parameter in their DaaS models.

978-0-7695-4635-3/11 $26.00 © 2011 IEEE
DOI 10.1109/SITIS.2011.14

context. Section III gives an introduction to query and mediated ontology (DO and COs) models serving as a basis for DaaS and mediation service models. Also, in this section we propose an extension to the DaaS model that allows automatic detection and resolution of data level conflict with the help of mediation services. Section IV gives details on the functioning of each component deployed in our proposal and the role of each one of them in query processing for DaaS composition. Section V describes details on the proposed conflict detection and resolution algorithm. Section VI gives a global view of our use case as an experimentation. Section VII explores related work on semantic heterogeneity detection and resolution during service composition based on query rewriting approach. Finally we summarize and discuss our results in Section VIII.

II. Motivation example

In this section, we provide an illustrating example from where the information needs of health actors are satisfied with a service oriented approach based on solution proposed by [2]. This approach raises up many problems, which motivate our proposal to apply semantic Web technologies to support mediation during DaaSs composition. Let us consider an e-health system exporting the set of DaaSs presented in Table I to query patient data. The description of DaaS can be seen in Table I, where the symbols “$” and “?” denote inputs and outputs of DaaSs, respectively. We assume that a physician submits the following real life query : “Q_1: check whether the medication identified by the code “801” to be prescribed to Joe with the PIN=80 [3] interacts with the ones currently taken by that patient”.

TABLE I
Example of DaaSs and Mediation Services

Service	Functionality	Constraints and DaaS provider
$S_{11}(\$x, ?y)$	Returns drugs y taken by a given patient x	DaaS provider is hospital1 and $y.code \in \{RxNorm\}$
$S_{12}(\$x, ?y)$		DaaS provider is hospital2 and $y.code \in \{NDC\}$
$S_{21}(\$x, ?y)$	Returns drugs y that interact with a given one x	$x.code \in \{RxNorm\}$ and $y.code \in \{RxNorm\}$
$S_{22}(\$x, ?y)$		$x.code \in \{NDC\}$ and $y.code \in \{NDC\}$
$S_3(\$x, \$y)$	Returns reference information y for drug x	$x.code \in \{ICD\}$
$S_{M1}(\$x, ?y)$	Returns drug code expressed in y code for given drug code expressed in x code	$x.code \in \{RxNorm\}$ and $y.code \in \{ICD\}$
$S_{M2}(\$x, ?y)$	Returns drug code expressed in y code for given drug code expressed in x code	$x.code \in \{NDC\}$ and $y.code \in \{ICD\}$

We assume for the moment that the physician will invoke automatically[4] the DaaS that provides the list of recent medication taken by Joe namely S_{11} or S_{12}. Then, he will invoke S_{21} and S_{22} to retrieve the list of drugs that interact with the drugs returned by S_{11} or S_{12} respectively. After that, he will invoke S_3 to retrieve more information about the drugs indicated by each interaction returned by S_{21} and S_{22}. However, as the system does not take into consideration the semantic conflict at data level, the physician need to invoke manually S_{M1} and S_{M2} to change the drug codes returned respectively by S_{21} (Rxnorm[5]standard) and S_{22}(NDC[6] standard) to codes acceptable by S_3 (ICD [7] standard). Thus, as the DaaS parameters use concepts with different semantics, the physician needs to manually select mediation service to solve semantic conflicts (Drug classifications, ontological reference, unit,....etc) and perform a meaningful composition as depicted in figure 1. The mediation steps remains new steps not considered in the previous solution when the client's context differs from the service providers.

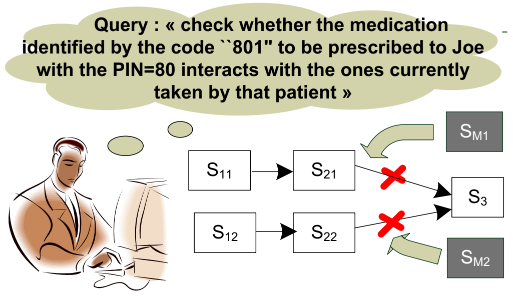

Query : « check whether the medication identified by the code ``801" to be prescribed to Joe with the PIN=80 interacts with the ones currently taken by that patient »
S11
S21
S12
S22
S3
SM1
SM2

Fig. 1. The automatic generated DaaS compositions and the manually invocation of mediation services

III. Models for ontology, services and query

This section gives details on the models adopted in our proposal.

A. Mediated ontology

Mediation Ontology expresses common entities and the relations among those entities. It can be visualized as a graph that contains nodes representing entities and edges representing relations among the entities. A mediated ontology, inspired from [2], [6], [7], includes two levels, namely, the Domain and the Contextual levels. The two levels have different namespaces for describing the domain concepts at the generic and contextual levels respectively as depicted in Figure 2. Such ontology should be defined by domain experts and specified using RDF/RDFS.

Definition 1 (Domain Ontology): An RDFS Domain Ontology is 6-tuple $<$ C, D, OP, DP, SC, SP$>$ where

[3]Patient Identification Number

[4]Querying mediated ontology allows DaaS discovering, after that they will be composed and executed according to the generated composition model.

[5]http://www.nlm.nih.gov/research/umls/RxNorm/

[6]The National Drug Code (NDC) is a unique product identifier used in the United States for drugs

[7]ICD: International Common Denomination

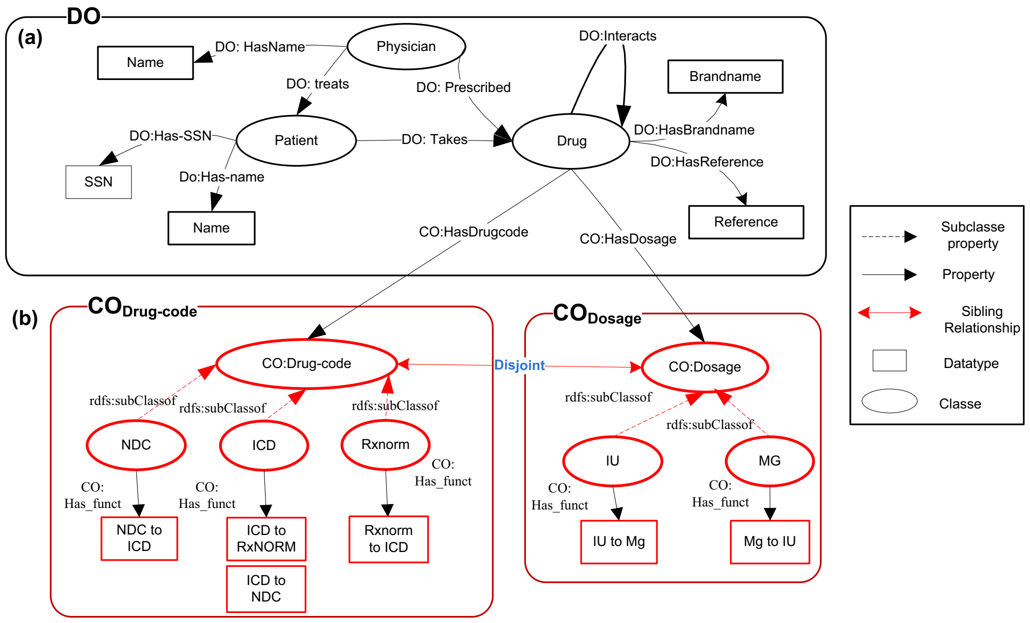


Fig. 2. (a) Domain ontology and (b) Contextual Ontologies

C is a set of classes; D is a set of data types; OP is a set of object properties; DP is a set of data type properties; SC is a relation over $C \times C$, representing the sub-class relationship between classes; SP is a relation over $(OP \times OP) \cup (DP \times DP)$, representing the sub-property relationship between homogeneous properties. Figure 2.(a) depicts the Domain Ontology, in which class nodes are represented by ovals and data type nodes are represented by rectangles[8].

Definition 2 (Contextual Ontology): An RDFS Contextual Ontology is 3 tuple $< C_g, C_i, \tau >$, where:

- C_g is a set of concepts that represent the different conflictual aspects of a generic concept in DO. Each C_g has a name and a set of sub concepts; the name represents a conflictual aspect of the associated generic concept. In the example depicted in Figure 2.(b), `CO:Drug-code` and `CO:Dosage` are C_g concepts.
- C_i is a distinct set of concepts having the same super-concept C_g. By definition, C_i are not allowed to have sub-concepts. The properties of C_i are defined as follows : Name of concept; Id is the property that represents the sequence number of a C_i concept among its siblings; A couple of properties reference the conversion functions between objects of C_i using their identifiers as references. The function name denotes the conversion from C_i to subsequent or precedent sibling, for instance `NDC-to-ICD` or `Rxnorm-to-ICD` as it follows the mapping direction. Supported conversions between sibling subclasses are $n \longrightarrow 1$ and $1 \longrightarrow 1$.
- τ refers to the sibling relationships on C_i and C_g. The relationships among elements of C_g is disjoint. However elements of C_i of a given C_g have peer relationship. They have similar data semantics, so that conversion or mapping can be performed among them.

Let us illustrate this definition with an example in Figure 2(b). The concept `DO:Drug` has a conflictual aspect called "code" that is described as a member of C_g in CO (i.e. $CO: Drug - Code$). The defined concept `CO:Drug-Code` can be represented differently in drug classifications, such as, $C_i = \{NDC, ICD, RxNorm, etc.\}$.

B. Conjunctive queries

In this paper we address conjunctive queries expressed using SPARQL, the do facto query language for the Semantic Web[9].

Definition 3: A conjunctive queries Q has the form: Q(X):-$< G(X,Y), C_q >$ where : $Q(X)$ is the head of Q, it has the form of relational predicate and represents the result of query; $G(X,Y)$ is the body of Q, it contains a set of RDF triples where each triple is of the form (subject. property.object); X and Y are called the distinguished and existential variables respectively, X and Y are subjects and objects in the RDF triples; $C_q = \{C1_q, C2_q, ..., Cn_q\}$ is a set of constraints expressed on X and Y variables in terms of traditional intervals or arithmetic expression like $x\theta constant$, $y\theta constant$ and where $\theta \in \{<, > \leq, \geq\}$. Formulated queries use concepts from DO ontology and properties from CO ontologies. Thus, a query can be seen as a graph with two types of nodes; class and literal nodes. Class nodes refer to classes in the

[8]More explication about the Domain ontology can be found in [2]

[9]SPARQL : http://www.w3.org/TR/rdf-sparql-query/

ontology. They are linked via object properties. Literal nodes represent data types and are linked with class nodes via data type properties. Figure 3 depicts the RDF graph of the query Q_1 described in our scenario.

```
Q ($w1,?Y1,$z1,?Y2):
?P  . Rdf:type . O:Patient
?P  . DO:hasSSN . "80"
?P  . DO:takes .?D1
?D1 . Rdf:type .DO:Drug
?D1 . CO:hasCode ?A
?A   . Rdf:type  CO:Drug-code
?A   . CO:codeValue  ?y1
?D1 .DO:intercats . ?D2
?D2 . Rdf:type .DO:Drug
?D2  .CO:hasCode ?C
?D2 .DO:hasreference ?y2
?C   .Rdf:type CO:Drug-code
?C   .DO:codeValue "801"
```

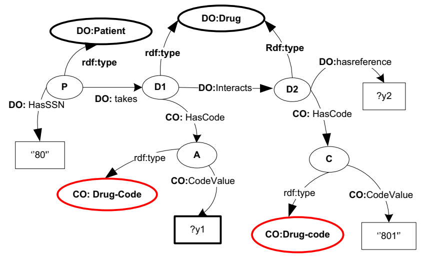


Fig. 3. Query in the running example

C. Extended DaaS model

We deem appropriate to follow the work of [2] to formalize the modeling of DaaS as PRV over a mediated ontology. As a DaaS is modeled uniquely over the entities of DO, it does not provides explicit semantics about its input and output parameters, so we extend its description with additional information describing more precisely how the semantics of the DO concepts are described according to the CO. Then, each DaaS model will be expressed as an adorned query [8]. The adornment is an annotation on variables, appearing in input and output parameters of a given DaaS and expressed in term of CO.

Definition 4 : The DaaS S_j is described as view in a Datalog-like notation over a DO and CO thus S_j model is $S_j(\$X_j, ?Y_j) :- < G_j(X_j, Y_j, Z_j), Co_j > |\alpha_{X_j}, \alpha_{Y_j}$ where: X_j and Y_j are the sets of input and output variables of S_j respectively; G_j represents the functionality of the DaaS which is described as a semantic relationship between input and output variables; Z_j is the set of existential variables relating X_j and Y_j; $Co_j = \{Co_{j_1}, ..., Co_{j_n}\}$ is a set of constraints expressed on X_j, Y_j or Z_j variables like $x\theta constant$ and $y\theta constant$ where $\theta \in \{<, > \leq, \geq\}$; α_{X_j} and α_{Y_j}, named adornment, are a set of RDF triplets describing the semantic (ontological reference, unit...etc) or domain expression of X_j and Y_j respectively. Each adornment α is indicated by the 2-tuple; $< C_g, C_i >$ where C_g: is a CO concept that represent the different conflictual aspects of X_j or Y_j; C_i is a subconcept's C_g.

Figure 4 gives an RDF view of the DaaS S_{11} depicted in Table 1 with an adornment depicted in red color.

D. Mediation service model

Mediation Services are also represented as a DaaS model (expressed in term of CO only) whereas their adornments are described as a set of RDF triples that define the conversion function between peers of $CO : C_i$ sub-concepts from the same $CO : C_g$ concept in a declarative way. We remind the reader that the different $CO : C_i$ are organized as an ordered list, hence a conversion from one to another is always a concatenation of conversion functions.

```
S11 ($z,?y) :-
(?P.rdf:type.O:Patient)
(?P.O:hasSSN.$z)
(?D. rdf:type.O:Drug)
(?P.O:takes ?D)
(?M.CO:HasCode.?C)
 (?C     rdf:type CO:Drug-code)
 (?C     rdf:type CO:RxNorm)
 (?C     CO:codeValue ?y)
```

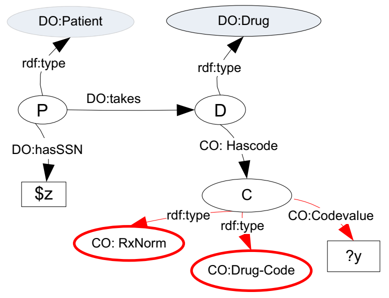


Fig. 4. DaaS model

Definition 5 : Mediation service S_j is modeled as below: $S_j(\$I_j, ?O_j) :- < G_j(I_j, O_j) > |\alpha_{Func<I_j, O_j>}$; Where $\$I_j$ and $?O_j$ defines the input and output parameter respectively required for using mediation service; $\alpha_{Func<I_j, O_j>}$ represents the conversion function from $CO : I_j$ to $CO : O_j$.

```
SM2 ($x,?z) :-
(?D  rdf:type          DO:Drug)
(?D  CO:hasCode     ?C)
(?C  rdf:type          CO:NDC)
(?C  rdf:type          CO:DrugCode)
(?C  CO:Codevalue  ?z)
(?D  CO:hasCode     ?A)
(?A  rdf:type           CO:ICD)
(?A  rdf:type           CO:DrugCode)
(?A  CO:Codevalue    $x)
(CO:NDC  CO:hasfunct  CO:NDC-to-ICD)
```

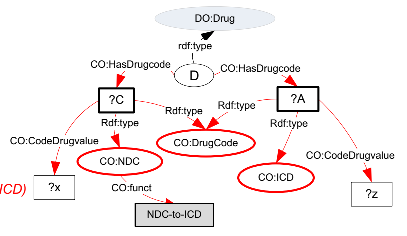


Fig. 5. Mediation service model

Figure 5 illustrates the RDF view of a mediation service S_{M2} utilized for converting a Drug-code from S_{22} to S_3.

IV. Architecture and Query Processing

In this section, our reference architecture is presented and supported by the role of each of its modules in the query processing for the composition and the mediation of DaaS, as shown in Figure 6.

A. Reference architecture

Data level: The lowest level of the architecture contains information stored in different components. **Service level**: The service level publishes the different services provided by several systems to different actors. This level provides two services categories. *DaaS services category:* provides specific data from databases or retrieve a document complaint model. *Mediation services category:* used mainly for mapping and converting the output parameter of a specific DaaS to the input parameter of a subsequent DaaS during service composition. These services advertise their WSDL [10] definitions into a service registry. For automatic discovery, selection, composition

[10]WSDL provides an XML-based grammar for describing a service interface

and mediation of service, the service registry includes a set of services descriptions (WSDL files) semantically annotated with *PRVs* expressed in term of mediated ontology. **Mediated level**: The mediated level is composed of two modules: *Mediated ontology*: The mediated ontology contains all the concepts and relations defined in domain. It will be used to annotate and query services (DaaSs, mediation services). We divide the ontology into two ontological levels which cuts the concept space into a Domain Ontology (DO) and a set of extensions named Contextual Ontologies (CO). *DaaS composition system* : Contains four sub-modules: the Service Locator Module (*SLM*), the Query Rewriting Module (*QRM*), the Conflict Detection and Resolution Module (*CDRM*) and the Query Plan Execution Module (*QPEM*). **Interface level**: The aim of this layer is to provide the interface for user whereby he can perform a query and receive results sets.

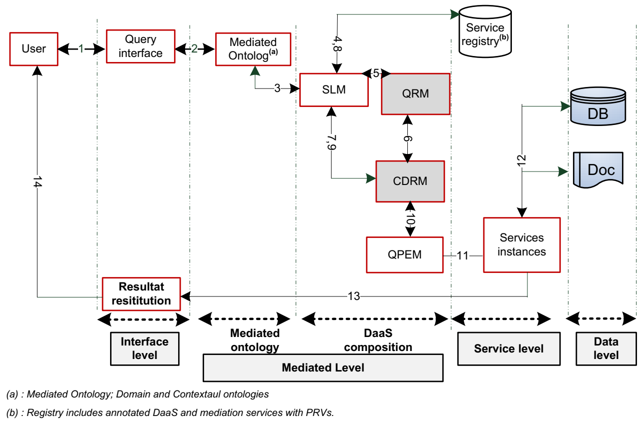


Fig. 6. Architecture and query processing for DaaS composition and mediation.

B. Query processing for Data-as-a-Services composition

The complete query processing steps are depicted in Figure 6. They include four processes. First, query formulation and service discovery, second, query rewriting, third, conflict detection and resolution, and finally query execution and result restitution. Firstly, **query formulation and service discovery** : In (1) and (2) the user issues SPARQL queries in terms of mediated ontology. Doing so, in (3) and (4) the *SLM* discovers DaaSs from the service registry that partially or completely matches the query entities (class nodes, object property nodes). Secondly, **query rewriting** : In (5), given a query *Q* and a set of DaaSs, the *QRM* rewrites *Q* as composition of DaaSs whose union of RDF graphs covers the RDF graph of *Q*. The composition query rewriting algorithm adopted in our work has two main phases: *Finding the covered query's subgraphs* and *Composition generation* as detailed in [2]. Thirdly, **conflict detection and resolution** : In (6), considering each combination generated by the *QRM*, which may encompasses semantic conflicts, *CDRM* tests any conflict by comparing output and input of subsequent DaaS in each query rewritings. The conflict is resolved with the insertion of a call to mediation services (7,8). Thus, in each DaaS combination, mediation services are added to resolve conflicts (9). Fourthly, **query execution and result restitution**: In (10, 11, 12), orchestrating the conflict-free composite service that has been generated requires a translation into an execution plan describing the data and control flows. Finally, (in 13 and 14) the QPEM synthesizes results and returns them to users through user interface.

Assuming for the moment that the DaaSs to be used has been found by a discovery process (3 and 4), such as by querying a mediated ontology the client must perform all of the actions cited above. Steps 1,2,3,4 and 5 have been implemented in systems that is based on query rewriting approach [2] to compose and execute DaaS composition model. Steps 6, 7, 8 and 9 achieved by *CDRM* are new steps required to do this when the clients context differs from the service providers.

V. Conflicts detection and resolution

In this phase we provide details about semantics conflicts detection and resolution in the DaaS combinations or rewritings generated by the *QRM*. Each rewriting will be passed in the *CDRM*. The CDRM is based on two-stage algorithm. The first stage (line 2 in algorithm 1) identifies the conflicts between the CO classes from subsequent DaaS in each rewriting. The conflicts are stored in a temporary Conflict Object set (COB) that stores mediation service parameters (input and output) and the position of the conflict. In case where conflicts are detected in any rewritings, the second component does the Conflict Resolution Module invokes the mediation service using the correspondences stored in COB (line 4 in algorithm 1).

Algorithm 1 Conflict Detection and Resolution algorithm

```
Require: R rewritings set, Meds mediation Service set.
Ensure: R' rewritings without conflict set.
1: for each $r \in \{R\}$ do
2:     $\mathcal{COB}$= Detection(r)
3:     if $\mathcal{COB} \neq \emptyset$ then
4:         R'= Resolution(r,$\mathcal{COB}$,Meds)
5:     else
6:         $R' = r$, No conflicts are detected in Rewriting r
7:     end if
8: end for
9: return $R'$
```

A. Conflict Detection

Conflicts arise when data elements that have to be exchanged between two interconnected DaaSs are interpreted differently. For this, as each DaaS service is adorned by CO concepts, we will compare each adornment for each interconnected DaaSs in each combination. For instance, let $CO : R_i$ and $CO : E_i$ be subclasses of the same conflictual class $CO : C_g$. Thus, if two interconnected DaaS Si and Sj having respectively in their RDF descriptions, the concepts

$CO : R_i$ and $CO : E_i$ as adornment of their input and output parameters, then, we have a semantic conflict of concept Cg_i. The semantic conflict type is a member of the set of conflictual concepts $C_g = \{Dosage, DrugCode, ..., etc\}$. Indeed, the *algorithm 2* is divided into two steps. The first step (line 1-3 in *algorithm 2*) takes each rewriting and iteratively verify the rule expressed previously for each parameter (adornment only) exchanged between interconnected services to find out all possible conflicts. The second step (line 4 in *algorithm 2*) stores conflict detected previously in the Conflicts-Objects set COB identified as 3 tuple $< O(S_i), I(S_j), index - position >$ where $O(S_i)$ is an adorned output parameter of a given DaaS source S_i , $I(S_j)$ is an adorned input parameter of a given DaaS target S_j, $Index - position$ indicates the position to which the service of mediation will be introduced into each combination.

Algorithm 2 Conflict Detection Algorithm

```
Require: r rewriting , $i, z \in \mathbb{N}$,
Ensure: $COB$ Conflict Object Set ,
1: for $i = 1$ to $n - 1$ do
2:     if $Output.S_i$ AND $Input.S_{j+1}$ have the same conflictual concept and different CO subclasses then
3:         $\mathcal{COB}_z$ = New conflict object( output.$\mathcal{S}_i$, input.$\mathcal{S}_{i+1}$, index-position(i+1)
4:         Add ($\mathcal{COB}$ , $\mathcal{COB}_z$ )
5:     end if
6: end for
7: return $\mathcal{COB}$
```

B. Conflict Resolution

In this stage, the *algorithm 3* will cross the list of the conflicting objects stored in COB for every rewriting and determines for every conflicting object, the appropriate mediation services allowing its resolution. A mediation service S_{Mn} allowing the resolution of a conflicting object $COBi$ between interconnected DaaS S_i and S_j is identified through:

- The input parameter$CO : O(S_i)$, which is the output of S_i;
- The output parameter$CO : I(S_j)$ which is the input of S_j;
- The conversion function as an adornment, defined as property of $CO : O(S_i)$ and targets $CO : I(S_j)$.

Once the mediation service is specified the *algorithm 3* will invoke it automatically from the Mediation service register in the rewriting according to index position stored previously. We deem appropriate to put a simplification hypothesis that each mediation service, in the register, resolves elementary conflict between subsequent CO_i concepts from the same C_g concept. Then, for each conflictual object $COBi$ of COB, the *algorithm 3* insert in each rewriting the mediation service allowing their resolution.

As a consequence, in the motivation example presented in section II, the mediation services S_{M1}(RxNorm-ICD) and

Algorithm 3 Conflict Resolution Algorithm

```
Require: $r$ rewriting with conflict, $\mathcal{COB}$ a set of Conflict Object, $S_{Mn}$ mediation service set, $i, j, k, z \in \mathbb{N}$.
Ensure: $r'$ rewriting without conflict.
1: for each $\mathcal{COB}z$ in $\mathcal{COB}$ do
2:     {according to conflict object identify mapping function ($Output.S_i$,$Input.S_j$) from CO ontology}
3:     ADD ($S_{Mn}$, $\mathcal{CO}i.indexe$) {ADD mediation service $S_{Mi}$}
4:     $COB_i.index = COB_i.index + 1$ {increment the index for the next mediation invocation}
5: end for
6: return r' {Rewriting without conflict}
```

S_{M2}(NDC-ICD) are added to the first and second DaaS compositions to resolve conflict as depicted in figure 1. Afterwards, the obtained conflict-free compositions will be translated into execution plans (i.e. orchestrations , represented as Directed Acyclic Graph)describing the data and control flows as explained in [2].

VI. Implementation

A prototype implementing the motivation example described in Section II [11] has been developed in Java 1.7 which can generate DaaS compositions that are executed in the open source GlassFish tools bundle for eclipse [12]. Our prototype illustrate how users can formulate different queries and how our system handles these queries to generate DaaS compositions without conflict.

Figure 7 presents the user interface of composition and mediation system. Users edit their queries in the *Query Editor*. The panel on the left-hand side, in one hand, gives a view of the DaaS and mediation services and in other hand, the DO and CO ontologies. Executing the query specified in the query editor results in the compositions shown in the "Compositions and mediation" tab. Jena-2.6.4 is used as the reasoning engine for RDFS [13].

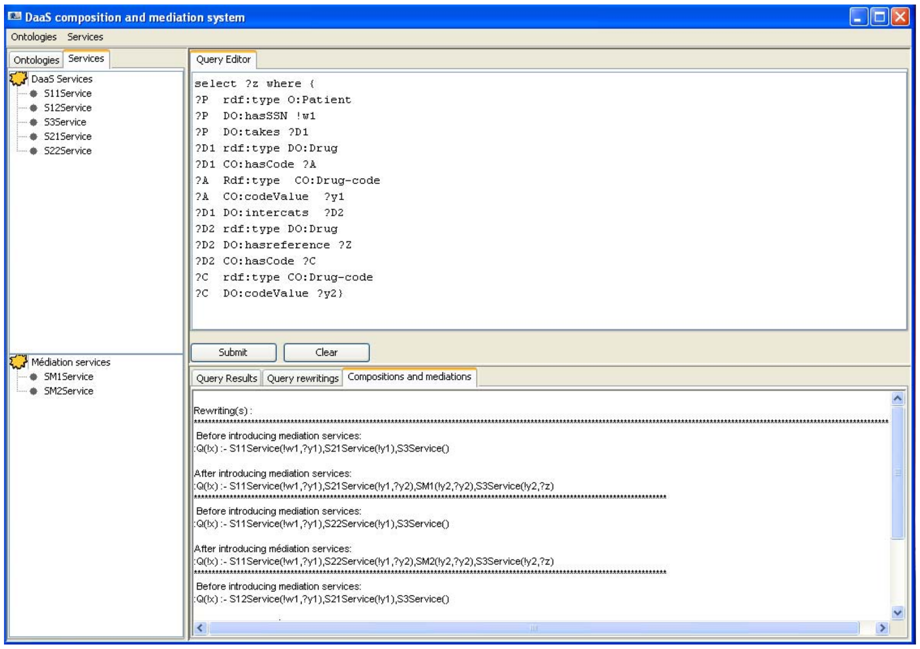
Also, we created the Domain and the Contextual ontologies with Protege tool 4.1[14]. We have used DO and CO ontologies to annotate the DaaS and mediation service description files (i.e. WSDLs) with an extended PRVs views as depicted in Figure 8 and Figure 9 respectively. The annotated files are then published to a Web service registry. Thus, by extending the parametrized RDF view by the CO concepts, the *CDRM* performs reasoning on the CO ontology to detect Conflicts within the generated rewriting. The *CDRM* invokes the mediated service which execute the predefined conversion or mapping function defined in CO. In sum, the set of conflict types identified in our solution is the set of conflictual concepts $CO : C_g$ (Drug-Code, dosage). However, other conflict types

[11] The implementation test is available in https://sites.google.com/site/drugimplementationtest/.

[12] http://dlc.sun.com.edgesuite.net/glassfish/eclipse/

[13] Jena Homepage http://jena.sourceforge.net

[14] http://protege.stanford.edu/



```
select ?z where {
?P  rdf:type O:Patient
?P  DO:hasSSN !w1
?P  DO:takes ?D1
?D1 rdf:type DO:Drug
?D1 CO:hasCode ?A
?A  Rdf:type  CO:Drug-code
?A  CO:codeValue  ?y1
?D1 DO:intercats  ?D2
?D2 rdf:type DO:Drug
?D2 DO:hasreference ?Z
?D2 CO:hasCode ?C
?C  rdf:type CO:Drug-code
?C  DO:codeValue ?y2}
```

```
Rewriting(s) :
*****************************************************************************************************
 Before introducing mediation services:
:Q(!x) :- S11Service(!w1,?y1),S21Service(!y1),S3Service()

After introducing mediation services:
:Q(!x) :- S11Service(!w1,?y1),S21Service(!y1,?y2),SM1(!y2,?y2),S3Service(!y2,?z)
********************************************************************
 Before introducing mediation services:
:Q(!x) :- S11Service(!w1,?y1),S22Service(!y1),S3Service()

After introducing médiation services:
:Q(!x) :- S11Service(!w1,?y1),S22Service(!y1,?y2),SM2(!y2,?y2),S3Service(!y2,?z)
********************************************************************
 Before introducing mediation services:
:Q(!x) :- S12Service(!w1,?y1),S21Service(!y1),S3Service()
```

Fig. 7. System Interface

can be added as $CO : C_g$ in order to resolve more semantic conflicts. The *CDRM* translated the DaaS compositions generated by *QRM* to produce DaaS compositions without any semantic conflict. Based on the prototype system, the results of experiment prove that the Service Composition and Mediation engine is feasible and effective [9].

```
  <? xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<service>
  <name>S21Service</name>
  <view>select ?y where {?D1 rdf:type Drug ?D1 CO:HasCode ?C ?C rdf:type CO:Drug-code
                              ?C rdf:type CO:NDC ?C CO:codeValue !x ?D1 interacts ?D2
                              ?D2 rdf:type Drug ?D1 CO:HasCode ?A ?A rdf:type CO:Drug-code
                              ?A rdf:type CO:RxNorm ?C CO:codeValue ?y}</view>
  <namespace>http://org.DaaS/</namespace>
  <Portname>S21Port</portname>
  <endpointadd>http://localhost:8080/DaaS/S21Service</endpointadd>
  <Inmsgtag>ResultsByOrder1</inmsgtag>
  <outmsgtag>ResultsByOrder1Response</outmsgtag>
</Service>
```

Fig. 8. DaaS annotation

```
  <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<service>
  <name>SM2Service</name>
  <view>select ?x where { ?D rdf:type DO:Drug ?D CO:hasCode ?C ?C rdf:type CO:NDC
                          ?C rdf:type CO:Drug-Code ?C CO:Codevalue ?z ?D CO:hasCode ?A
                          ?A rdf:type CO:RxNorm ?A rdf:type CO:Drug-Code ?A CO:Codevalue
                          $x ?CO:NDC CO:hasfunct ?CO:RxNorm }</view>
        <namespace>http://Services/</namespace>
        <portname>SM2Port</portname>
        <endpointadd>http://localhost:8080/MediationServices/SM2Service</endpointadd>
        <inmsgtag>LoincToSnomed</inmsgtag>
        <outmsgtag>LoincToSnomedResponse</outmsgtag>
  </service>
```

Fig. 9. Mediation service annotation

VII. Related Work

This section explores related work on both query rewriting techniques and semantic heterogeneity detection and resolution for service composition.

Many works have proposed solution for automatic Web service composition approaches based on query rewriting approaches [2], [3], [4], [5]. Also these works are closest to ours since all of these are targeted at DaaS services and explicitly regard DaaS as Parametrized RDF views (SPARQL queries) and transform the service composition problem into a query rewriting problem. However, we can say that DaaS composition systems proposed by these works don't have faced the data level heterogeneities at semantic level; unlike to our work who proposes a service oriented approach to resolve data level conflict by extending the PRVs service model for DaaS and mediation service.

Besides, most of the efforts in Web services composition focus on automatically constructing the workflow logic by means of ontologies, but only a few approaches have been developed to handle semantic heterogeneity in Web services composition. In this sense, [10], [11], [12], [13] has investigated the data-level heterogeneity between Web services through mapping relations to establish the direct correspondence between the messages of two services. Also these works require Input-Output service parameters to be annotated with classes from the Domain ontology as semantics, which is too rigid. However, unlike to our approach these approaches are restricted to simple composition scenarios in which only two services are integrated.

Another set of work which is very related to our efforts in the area of context representation for web service composition. The approaches discussed in [14] and [15] propose a lightweight ontology which needs a small set of generic concepts to capture contextual semantics. Unlike to [14], [15] proposes solution where the data mediation is achieved by only a mediator (external) Web service inserted between the service during the composition execution. These services are semantically described using the WSDL extensibility elements and a domain ontology to which are associated the contextual ontologies. Unlike to [15] proposition where the notion of context is defined as a semantic object, we define declaratively the context as the extension of an PRV (adornment).

VIII. Conclusion

In this paper, the DaaS service composition proposed in [2] has been extended to handle the semantic conflict based on a mediation service approach. Indeed, a two layers mediated ontology has been proposed to extend the DaaS model and to define a mediation services. The loosely coupled aspect of our approach, allows keeping mediation concerns orthogonal from functionalities of DaaS.

We have also performed preliminary evaluation that showed satisfactory results. In the future, we plan to define an inter-context semantic model to consider the detection and the resolution of complex semantic conflict.

References

[1] W. Zhao, C. Liu, and J. Chen, "Automatic composition of information-providing web services based on query rewriting," *SCIENCE CHINA Information Sciences*, pp. 1–17, 2010, 10.1007/s11432-011-4341-5.

[2] M. Barhamgi, D. Benslimane, and B. Medjahed, "A query rewriting approach for web service composition," *IEEE Transactions on Services Computing*, vol. 3, pp. 206–222, 2010.

[3] L. Zhou, H. Chen, J. Wang, and Y. Zhang, "Semantic web-based data service discovery and composition," *Semantics, Knowledge and Grid, International Conference on*, vol. 0, pp. 213–219, 2008.

[4] J. Lu, Y. Yu, and J. Mylopoulos, "A lightweight approach to semantic web service synthesis," in *Web Information Retrieval and Integration, 2005. WIRI '05. Proceedings. International Workshop on Challenges in*, 2005, pp. 240 – 247.

[5] R. Vaculín, H. Chen, R. Neruda, and K. Sycara, "Modeling and discovery of data providing services," in *Proceedings of the 2008 IEEE International Conference on Web Services*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 54–61. [Online]. Available: http://portal.acm.org/citation.cfm?id=1474549.1474834

[6] S. Ram and J. Park, "Semantic conflict resolution ontology (scrol): an ontology for detecting and resolving data and schema-level semantic conflicts," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 16, no. 2, pp. 189 – 202, feb. 2004.

[7] Q. Liu, T. Huang, S.-H. Liu, and H. Zhong, "An ontology-based approach for semantic conflict resolution in database integration," *Journal of Computer Science and Technology*, vol. 22, pp. 218–227, 2007, 10.1007/s11390-007-9028-4. [Online]. Available: http://dx.doi.org/10.1007/s11390-007-9028-4

[8] D. Calvanese, G. D. Giacomo, M. Lenzerini, D. Nardi, and R. Rosati, "A principled approach to data integration and reconciliation in data warehousing," in *In Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW99*, 1999.

[9] A. I. Amine, B. Djamal, B. Mahmoud, M. Mrissa, and A. Zaia, "Electronic health record daas services composition based on query rewriting," *Transactions on Large-Scale Data and Knowledge-Centered Systems*, vol. 4, pp. 95–123, 2011.

[10] B. Spencer and Y. Liu, "Inferring data transformation rules to integrate semantic web services," in *In International Semantic Web Conference*. Springer, 2004, pp. 456–470.

[11] D. Gagne, M. Sabbouh, S. Bennett, and S. Powers, "Using data semantics to enable automatic composition of web services," in *Services Computing, 2006. SCC '06. IEEE International Conference on*, sept. 2006, pp. 438 –444.

[12] M. Sabbouh, J. L. Higginson, C. Wan, and S. R. Bennett, "Using mapping relations to semi automatically compose web services," in *Proceedings of the 2008 IEEE Congress on Services - Part I*, ser. SERVICES '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 211–218. [Online]. Available: http://dx.doi.org/10.1109/SERVICES-1.2008.12

[13] M. Nagarajan, K. Verma, A. Sheth, and J. Miller, "Ontology driven data mediation in web services," *International Journal of Web Services Research*, vol. 4, no. 4, pp. 104–126, 2007.

[14] X. Li, S. Madnick, H. Zhu, and Y. Fan, "Reconciling Semantic Heterogeneity in Web Services Composition," *ICIS 2009 Proceedings*, p. 20, 2009.

[15] M. Mrissa, C. Ghedira, D. Benslimane, and Z. Maamar, "A context model for semantic mediation in web services composition," in *Conceptual Modeling - ER 2006*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2006, vol. 4215, pp. 12–25.