DMaaS : Syntactic, Structural and Semantic Mediation for Service Composition

Mohamed Sellami

ASCOLA group, INRIA,
École des Mines de Nantes, France,
firstname.surname@mines-nantes.fr*
*Corresponding author, work started while working at Telecom SudParis.

Pierre De Vettor, Michael Mrissa, Djamal Benslimane

CNRS, Université de Lyon,
LIRIS UMR 5205, Université Lyon 1, France,
firstname.surname@univ-lyon1.fr

Bruno Defude

CNRS, Telecom SudParis,
Samovar UMR 5157, Evry, France,
firstname.surname@it-sudparis.eu

Abstract: Service composition is a major advance service-oriented computing brings to enable the development of distributed applications. However, the distributed nature of services hampers their composition with data heterogeneity problems. In this paper, we address these problems with a decentralized Mediation-as-a-Service architecture that solves data inconsistencies occurring during the composition of business services. As an extension to our previous work that focused on data interpretation problems, we present in this paper a solution to solve data inconsistencies at the syntactic, structural and semantic levels. We show how syntactic, structural and semantic mediation techniques can be combined, and how semantic mediation provides useful information that helps structural and syntactic mediation. We demonstrate how our architecture enables decentralized publication and discovery of mediation services. We motivate our work with a concrete scenario and validate our proposal with experiments.

Keywords: Web services; Composition; Data Mediation; Data Integration; Data Semantics.

Reference to this paper should be made as follows: Sellami, M., De Vettor, P., Mrissa, M., Benslimane, M. and Defude, B. (xxxx) 'DMaaS : Syntactic, Structural and Semantic Mediation for Service Composition', *Int. J. Autonomous and Adaptive Communications Systems*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Mohamed Sellami received the PhD degree in computer science from Telecom SudParis in 2011. He is a Postdoctoral researcher at the ASCOLA group part of the computer sciences department of Ecole des Mines de Nantes in France. Before joining ASCOLA, he was also a postdoctoral researcher at the SIMBAD group part of the computer sciences department of Telecom SudParis. His current research areas include service oriented computing, and cloud computing. More information can be found at: http://www.emn.fr/z-info/sellami/.

Pierre De Vettor is a PhD student at the University of Lyon. He is preparing his PhD with the Audience Labs company based in Lyon and the LIRIS laboratory. His main research interests include service oriented architectures, data mediation and the semantic Web. His thesis subject looks at how to provide linked data integration for raw data coming from heterogeneous data sources.

Michael Mrissa is Associate Professor of Computer Science and member of the LIRIS CNRS Laboratory, Claude Bernard Lyon 1 University, France. He received his PhD from the Claude Bernard Lyon 1 University in fall 2007. His main research interests are related to Web services, data and information management, and the semantic Web. His publication list includes international journals and conferences such as ACM TOIT and IEEE TSC and he has served on numerous conference program committees.

Djamal Benslimane is professor of computer sciences at Claude Bernard Lyon 1 University and Co-head of SOC research team of the Lyon Research Center for Images and Intelligent Information Systems (`http://liris.cnrs.fr/`) in Lyon, France. His research interests include Distributed Information Systems and Web services. Djamal Benslimane has published several papers in well-known journals (e.g. Communications of the ACM, ACM Transactions On Internet Technology, ACM Transactions on Software Engineering and Methodology, IEEE Transactions on Service Computing, IEEE Transactions on Systems, Man, and Cybernetics, IEEE Internet Computing, Data & Knowledge Engineering).

Bruno Defude received the Ph.D. degree in 1986 and Habilitation degree in 2005 in computer science, respectively, from Grenoble INP and Paris VI University, France. He is currently a Professor and Head of the Computer Science Department at TELECOM SudParis, France. His research interests are distributed data management, cloud data management, and semantics for B2B integration. He was the Chair of the IEEE WETICE Conference in 2011. He has been participating in several national and European research projects. He has published more than 50 research papers in international conferences and has served as program committee member in many conferences and workshops.

1 Introduction

The development of service-oriented computing (SOC) has been promoting remote software interactions inside and across organizational boundaries. In particular, Web services are interoperable components that rely on XML-based languages and protocols to be invoked. The task of service composition consists in connecting services to each other in a workflow so that they exchange data via their input and output parameters, to provide value-added functionality.

However, the distributed nature of services raises several problems that hamper their widespread adoption. In this paper, we are interested in data heterogeneity problems raised during service composition. Data heterogeneity problems occur along three levels: syntactic, structural and semantic. At the syntactic level, low level incompatibility may cause parsing problems between software applications (i.e. JSON or XML syntax). The use of XML as a common syntax solves data inconsistencies, but competing languages such as JSON may raise such syntactic heterogeneity that hamper service interactions. At the structural level, despite the fact that services are semantically described, problems occur when data needs to be merged or splitted according to different schemas. At the semantic level, data values must be attached to ontology concepts, and the additional information that allows their correct interpretation must be explicitly described as well. In addition, although approaches to deal with data heterogeneity problems have been proposed, with mediators in WSMO [1] or ESB [2, 3], the search for a scalable, decentralized mediation solution remains crucial, as centralized solutions raise scalability and performance problems on the Web.

In this paper, we develop an architecture that promotes Mediation-as-a-Service (MaaS) to boost the integration of *mediation services* into service-oriented computing. Mediation services are Web services dedicated to data conversion. Our architecture relies on a distributed hash table that enables the decentralized publication and discovery of mediation services, allowing service providers to offer mediation services to their clients, besides business services. We introduce the Decentralized Mediation as a Service (DMaaS) approach to handle the three data heterogeneity levels and facilitate data exchange between heterogeneous services. While previous work [4] focused on the data interpretation problem at the semantic level only, this paper extends our approach to the structural and syntactic levels, and demonstrates how mediation at different levels can be combined in the same framework.

This paper is organized as follows. Section 2 shows the need for mediation and illustrates our proposal with a motivating scenario. Section 3 explains how we model data semantics and context with the help of domain and conflictual aspect ontologies. Section 4 presents the notion of mediation services as central components of our architecture. Section 5 shows how data inconsistencies that appear in workflows are detected and resolved at runtime. It explains how the invocation of mediation services during the execution of a workflow is triggered. Section 6 proposes algorithms to publish and discover mediation services distributed over the network. Section 7 describes the prototype we developed as a proof of concept to our work. Section 8 discusses related work and shows the advantages of our approach and Section 9 discusses our results and presents directions for future work.

2 Challenges and Motivating Scenario

Our scenario is inspired from the SWS Challenge scenario[a] which has been developed around a typical purchase order use case, where composed Web services show data heterogeneity problems. In this scenario, Web services that present data heterogeneities are composed and the participants to the challenge aim at providing a mediator component that solves these data heterogeneities and enables valid communications via data mappings. Figure 1 illustrates the purchasing goods workflow we use as motivating scenario.

A customer, through the Send PO (Purchase Order) operation, contacts the CRM (Customer Relationship Management) and OM (Order Management) Web services of a company to ensure a goods purchase. We identify the following differences for the data values to be exchanged between the services involved in this scenario.

- At the syntactic level:
  - operations offered by the customer Web services (i.e. *Send PO* and *Receive PO Confirmation*) use XML as a data syntax whereas the company Web services use JSON. The piece of information from customer and company needs to be translated into the right syntax for correct data exchange.

[a] http://www.sws-challenge.org/wiki/index.php/Scenario:_Purchase_Order_Mediation

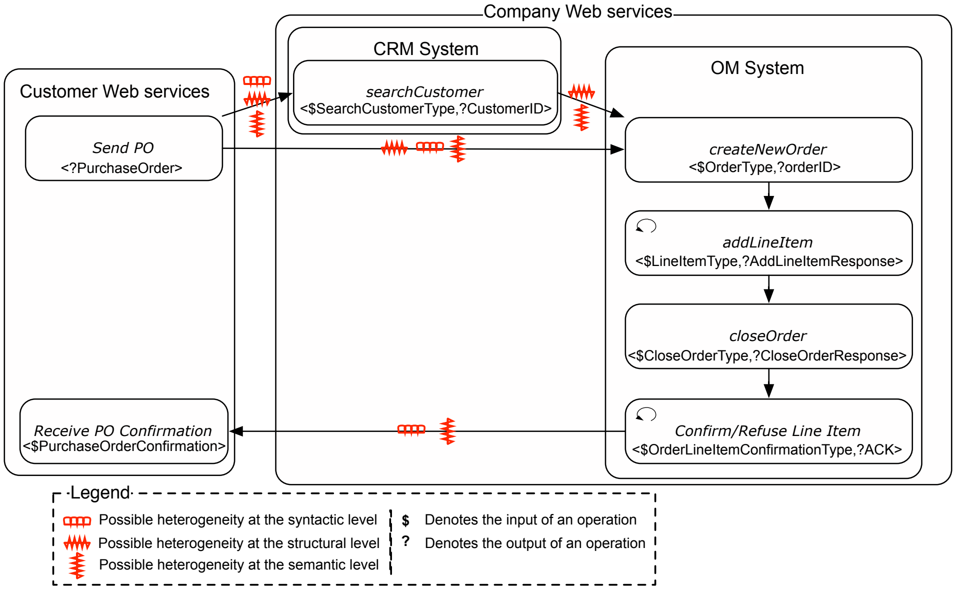

Company Web services
CRM System
searchCustomer
<$SearchCustomerType,?CustomerID>
OM System
Customer Web services
Send PO
<?PurchaseOrder>
createNewOrder
<$OrderType,?orderID>
addLineItem
<$LineItemType,?AddLineItemResponse>
closeOrder
<$CloseOrderType,?CloseOrderResponse>
Receive PO Confirmation
<$PurchaseOrderConfirmation>
Confirm/Refuse Line Item
<$OrderLineItemConfirmationType,?ACK>
Legend
Possible heterogeneity at the syntactic level
Possible heterogeneity at the structural level
Possible heterogeneity at the semantic level
$ Denotes the input of an operation
? Denotes the output of an operation

Figure 1: Purchase order scenario

- At the structural level:
  - the output `PurchaseOrder` of the *Send PO* operation contains different elements related to the requester and his order (name, companyName, deliveryAddress, productID, etc.). This data has to be adapted to the input `SearchCustomerType` (containing only the companyName) required by the *searchCustomer* operation of the CRM Web service.
  - the input `OrderType` for the *createNewOrder* operation of the OM Web service has a complex structure composed of several elements to be provided by `PurchaseOrder`, the output of the *Send PO* operation, which is defined according to a different structure. Similar heterogeneities are also identified for the inputs of the *addLineItem*, *closeOrder*, *confirm/refuse Line Item* and *receive PO Confirmation* operations.
- At the semantic level:
  - phone numbers and postal codes (number interpretation)
  - dates and timestamps (interpretation of ordering and format)
  - prices (interpretation of currency and VAT rate)

This scenario illustrates the challenges addressed in this paper. Firstly, there is a need to automatically detect and solve the data inconsistencies that occur in a workflow. These inconsistencies occur at the syntactic, structural and semantic levels, it is required to take each level into account for accurate data mediation. Secondly, there is a need to develop a generic solution based on mediation components to resolve these inconsistencies. In the following, we propose a solution based on Web services. Thirdly, there is a need to deploy scalable mechanisms for the decentralized discovery and selection of these mediation components. Our solution relies on a Chord ring to answer this need.

3 Conflictual Aspects for Unambiguous Data Interpretation

Service composition implies organizing the invocation order of the different services, so that the data produced by one service can be reused as input to another service (called a data dependency). The invocation order and data dependencies between services are typically represented as a workflow (we present a simple workflow language in Section 5.1).

Semantic and structural conflicts occur when data sent from a service to another are not interpreted correctly. We identify two concerns that should be addressed to solve semantic and structural conflicts in a workflow. First, it is required to make explicit the domain knowledge the workflow is bound to. Domain knowledge is typically described with the help of domain ontologies. The second concern relates to describing how to interpret concepts of domain ontologies. Indeed, concepts can be understood according to different interpretations[b] leading to structural and/or semantic conflicts, and their automatic reconciliation is a difficult task.

To automate data conversion, there is a need to explicitly describe the computation required to, on the one hand, convert a data value from one interpretation to another, and on the other hand to adapt data structure from a format to another. To do so, we rely on the notion of context, defined as *the collection of implicit assumptions that are required to perform a correct interpretation of data* [5] and on conflictual aspect ontologies (CAO) [6] to make explicit the different context dimensions that are necessary to allow unambiguous interpretation of domain concepts.

In this section, we introduce the notion of conflictual aspect ontology, explain its role and how it allows us to define the context of execution with respect to data mediation. We also explain how we annotate services with domain and contextual aspect ontologies.

3.1 Domain and Conflictual Aspect Ontologies

Our organization into domain ontologies (DO) and conflictual aspect ontologies (CAO) applies Dijkstra's *separation of concerns* principle [7] to describe the semantics of data. DOs describe the different concepts of a knowledge domain and their relationships. They are useful for comparing concepts and inferring semantic compatibility between a pair of concepts, for example with subsumption relationships. Being able to match concepts means that the values exchanged between services refer to conceptually compatible entities. This step is necessary but not sufficient to reach semantic-level interoperability. CAOs are dedicated to capturing the variety of information required for the correct interpretation of a specific DO concept instance. CAOs are referred to as "conflictual" as they capture the different elements that make the interpretation of a concept instance vary from one service to another (and create conflicts). We define a CAO as a triple $< Ac_g; Ac_i; \tau >$, where:

- Ac_g is a set of classes which represents the different conflicting aspects of a DO entity. Each class $ac_g \in Ac_g$ has a set of sub-classes. Each ac_g class has a name representing a conflicting aspect, such as, "cao:Currency".

- Ac_i is a distinct set of classes having one super-class in Ac_g. By definition, a class $ac_i \in Ac_i$ is not allowed to have sub-classes. For instance "cao:Euro" and "cao:Dollar" are two classes from the "cao:Currency" class.

[b] For example, the concept of price may refer to different currencies and the concepts of length or weight are understood according to different units.

- τ refers to the sibling relationships on Ac_i and Ac_g. The relationships among classes in Ac_g are "disjoint". However, elements of Ac_i for a given ac_g are related by the "sameCA" property which indicates that they describe the same context dimension.

3.2 Describing Context

3.2.1 Defining Context with Conflictual Aspect Ontologies

We define a context $Ctxt$ as a set of $ac_i \in Ac_i$ where ac_i is the instance that characterizes the interpretation of a dimension $ac_g \in Ac_g$. The interpretation of a single DO instance is made explicit with the help of several context dimensions described with disjoint Ac_g, which are in turn instantiated with ac_i in CAOs. It is possible to know the context dimension from a given ac_i by looking at the superclass Ac_g of the classAc_i it instantiates.

Fig. 2 gives a partial overview of the domain and conflictual aspect ontologies developed in our scenario. The "currency" and "VATRate" dimensions are attached to an instance of the "Price" DO concept, and the different dimension instances form the "context" that describes a specific interpretation of a price instance.

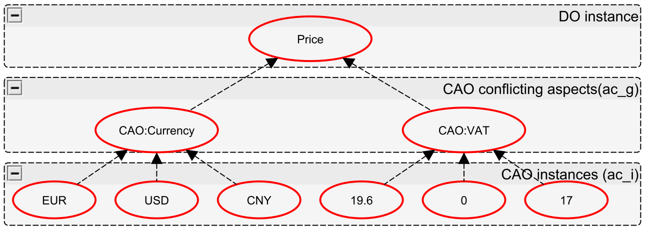


Figure 2: Excerpt of a CAO Ontology

During the execution stage of a service composition, it is necessary to make explicit the context dimensions attached to the DO instances that services exchange and to enable semantically meaningful, unambiguous data exchange by transforming data instances according to the different contexts of involved services. Indeed, the context dimensions used to describe data may vary from a service to another. In a situation where a DO instance is described according to different dimensions, mediation is possible according to the set of shared dimensions. Thanks to Web service annotations, shared dimensions are identified using the "sameCA" property that links Ac_i to each other in the conflictual aspect ontology. Mediation services, to be described in Section 4, allow data mediation between elements that belong to shared context dimensions.

3.2.2 Annotating Services

Traditional SOAP Web services can be annotated using SAWSDL [8] with the DO instances attached to their input and output parameters. We annotate service descriptions using the MSM model [9] that partially reuses the standard SAWSDL annotation and is able to describe RESTful services too. We use the `modelreference` attribute to attach a URI to the DO instance that describes the I/O parameter with a set of contextual dimensions defined in CAO ontologies. Indeed, our annotation incites service providers to collaboratively develop the DOs and CAOs that correspond to the business services they provide and extend their service descriptions. Several DO are available on the Web, such as Dublin Core or at `http://schema.org`. CAOs are specific to the local interpretation of service providers and therefore should be designed in accordance with the corresponding mediation services. These latter convert data along the context dimensions that are made explicit with CAO properties. Ideally, each mediation service implements a data conversion from a context dimension to another, thus making possible the conversion from any context dimension to any other.

4 Mediation Services

The idea to develop mediators as services, firstly introduced in the WSMO architecture [1], offers several advantages, such as loose coupling, easy reuse, scalability and composition. In our work, we introduce three types of mediation services: Syntactic Mediation Services, Structural Mediation Services and Conflictual Aspect Mediation services that respectively solve data inconsistencies at the syntactic, structural and semantic levels.

Syntactic Mediation Services Syntactic-level mediation is performed with simple services that convert from a syntax to another. In our scenario, we developed a simple XML-to-JSON conversion service as these are two famous languages used nowadays.

Structural Mediation Services Structural mediation in the data flow of a Web services composition is ensured through specific WSs that enable data mapping (DM) [10]. We identify four types (1-1, 1-N, N-1 and N-M) for a structural mediation service according to the elements handled by the mapping. In these types, '1' denotes simple elements and 'N/M' denotes complex elements.

- **1-1**: this is the case of mapping a simple element into another simple element.
- **1-N**: this is the case of mapping a simple element into a complex element. For example, a 1-N data mapping can be a split. Using a split data mapping, a *deliveryAddress-1* represented as a string "1 Ave des Champs Elysees, 75008 Paris" can be mapped to a *deliveryAddress-2* data type as a tuple <1,Ave des Champs Elysees, Paris, 75008>.
- **N-1**: this is the case of mapping a complex element into a simple element. For example, an N-1 data mapping can be a merge. Using a merge data mapping, a *deliveryAddress-2* can be mapped to a *deliveryAddress-1* data type.
- **N-M**: this is the case of mapping a complex element into another complex element. For example, an N-M data mapping can convert a *deliveryAddress-2* represented by a tuple <1,Ave des Champs Elysees, Paris, 75008> to a *deliveryAddress-3* data type as another tuple <1,Ave des Champs Elysees, 75008, Paris>.

Conflictual Aspect Mediation Services Semantic mediation is ensured through another kind of mediators called conflictual aspect mediation services (CA mediation services). These services provide a unique operation converting the data from one representation (i.e. according to one context dimension) to another and thus solving data interpretation inconsistencies.

Mediation services are created by Web services providers. Creating these services can be intuitive while publishing a new Web service. For example, while providing an order management service, the service provider can publish at the same time a DM service for delivery address structure transformation or a CA mediation services for currency conversion.

4.1 Mediation Service Description

Mediation services are stateless (i.e. no preconditions and no effects) and there is no need to represent a relationship between an input and an output of a mediation service since the only relation linking them is a "*convertTo*"-like relation. Thereby, SAWSDL [8] is adequate for describing a mediation service and more precisely the structural or semantic conversion offered by specifying its inputs and outputs. In Listing 1, we give as example an excerpt of the SAWSDL description of a DM service offering the structural mediation to transform a delivery address from one representation to another (see Section 4).

Listing 1: Excerpt of a DM Web service SAWSDL description

```
<types>
 <element name="address" type="taddress" modelReference="O1.owl#address"/>

 <complexType name="taddress">
  <sequence>
   <element name="streetNumber" type="int"/>
   <element name="streetName" type="string"/>
   <element name="city" type="string"/>
   <element name="postalCode" type="int"/>
  </sequence>
 </complexType>

 <element name="adresse" type="string" modelReference="O2.owl#adresse"/>
</types>

<interface name="convert">
 <operation name="convertTo">
  <input element="adresse"/>
  <output element="address"/>
 </operation>
</interface>
```

4.2 Mediation Service Operation

When a mediation service is invoked, it means that the workflow is being executed, that a syntactic, structural or semantic conflict has been detected and that the appropriate mediation service has been inserted to intercept and convert data. These aspects are dealt with in the next sections of the paper. A conflictual aspect mediation service should receive an input value with the conflictual aspect it takes as input and the conflictual aspect in which it should return data. Each mediation service knows how to convert data for a specific context element (for example, "currency" converter or "delivery address" converter). These mediation services take as input a data value to be converted, a source context dimension describing the piece of data and a target context dimension. For instance, the following values are sent to a CA mediation service for currency conversion : “18” (data value, instance of the “do:currency” class), “cao:EUR” (source context), “cao:USD” (target context). In this example, “cao:EUR” and “cao:USD” are instances of a CAO. The CA mediation service returns the data value in the target context (“24” for instance). Accordingly, a structural mediation service receives input data according to the source structure and sends transformed data according to the target structure, and a syntactic mediation service converts data from a syntax to another.

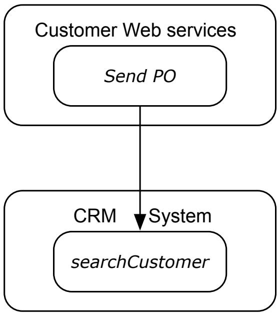
5 Detection of Data Conflicts

In this section, we introduce a simple workflow language to represent data dependencies between services that participate in the same composition and show how our workflow execution engine detects and resolves semantic conflicts.

5.1 Workflow language

In order to perform mediation-enabled execution of a service workflow and insert mediation services into existing workflows, we represent the latter as a graph with services as vertices and data connections as edges. We rely on a simple XML-based data flow-oriented workflow language, inspired by SCUFL [11] and MoML [12]. We relied on this language for its simplicity that avoids the complexity of languages such as BPEL [13]. Our language relies on a set of `service` and `link` elements that respectively describe services and the data flows that connect them. The `service` element must contain a unique `id` attribute for identification and a unique `description` child element that contains the URL of the service description file. The `description` element is extended with a `format` attribute that allows automatic parsing (format could be SAWSDL, OWL-S [14], MSM [15], etc.). These elements identify services and provide the means to enable their invocation from their description files. The `link` element contains an `id` attribute plus `source` and `target` child elements that contain the IDs of services involved in the link. Data flows from the `source` to the `target` service. The execution of a workflow starts with the services that do not participate as `target` in links and naturally follows the links to other services.

In Figure 3, we provide as an example an excerpt of the workflow description associated to our purchase order scenario (Figure 1) expressed in our data flow-oriented language. This code illustrates the data flowed between the customer Web service, through the *Send PO* operation, and the CRM Web service, through the *searchCustomer* operation. The services are represented by the `<wf:service id="SendPO">` and `<wf:service id="SearchCustomer">` XML elements and identified, using the *id* attribute, by their respective operation names. Their associated SAWSDL descriptions are referenced in the `<wf:description format="sawsdl">` elements. The `<wf:link id="SendPOtoSearchCustomer">` element represents the link between both services: `SendPO` as the source service and `SearchCustomer` as the target.



```
<?xml version="1.0" encoding="UTF-8"?>
<wf:workflow id="purchase" xmlns:wf="http://
soc.univ-lyon1.fr/dmaas/files/workflows.xsd#">

   <wf:service id="SendPO">
      <wf:description format="sawsdl">
       http://soc.univ-lyon1.fr/DMaaS/customer.sawsdl
      </wf:description>
   </wf:service>

   <wf:service id="SearchCustomer">
      <wf:description format="sawsdl">
       http://soc.univ-lyon1.fr/DMaaS/crm.sawsdl
      </wf:description>
   </wf:service>

   <wf:link id="SendPOtoSearchCustomer">
      <wf:source>SendPO</wf:source>
      <wf:target>SearchCustomer</wf:target>
   </wf:link>
</wf:workflow>
```

Figure 3: Excerpt from the purchase order scenario.

5.2 Conflict detection for each type of conflict

Each type of conflict is detected at a different time, according to its abstraction level. The first type of conflict to be tracked is the semantic conflict for 1-to-1 links, because these conflicts are not dependent from other conflicts. The second tracking to be performed concerns structural conflicts, which allows us to detect the 1-to-N, the N-to-1 and the N-to-N structural conflicts from the matching concepts. Semantic conflict detection is then performed for each element of these complex mappings. The system then tracks syntactic conflicts between matching elements.

5.2.1 Semantic conflicts

Thanks to our workflow language, the detection of semantic conflicts is simple to perform. For each message part involved in a `link` element of the workflow, we must make sure that the output concept from the service identified by the `source` element and the input concept from the service identified by the `target` element match[c].

Once concept matching has been realized, conflictual aspects over context dimensions are evaluated by looking at the CAO instances that are attached to output A and comparing their values with those of B. The result of this second step is the input to the mediation service discovery algorithm developed in Section 6.2, which looks for conversion services that enable conversion from a context to another. The detection of data heterogeneity in the data flow is performed as follows. We developed a function called `detectConflicts` that runs through all the `link` elements of the workflow. The function retrieves the description files of services involved in the link. It extracts via our annotation the DO concept instances and makes sure that they match. For each couple of matching concepts, another function called `getContextHeterogeneity` fetches the CAO ontologies that connect context dimensions -and their instances- to the data concepts. The function returns a set of mediation

[c] Here, matching means that B subsumes A where A is the output from a source service and B is the input from a target service.

needs according to the set of common dimensions the concepts share. For example, instances of the price DO concept may requires EUR to JPY conversion according to their currency values.

5.2.2 Structural conflicts

In our context, the detection of structural conflict is performed in a second time, after the semantic conflicts detection, we search through all the input and output of our workflow representation and detect each parameter which has not been connected. The analysis includes the following steps. A first step detects N-to-1 and N-to-N structural conflicts from the list of input concepts left after the semantic detection. A second step searches within the output list for each output concepts which does not have connection to detect the 1-to-N structural conflicts. For each input and output, the system retrieves the concepts involved in the link. It extracts via our annotation the DO concept instances and check whether or not, this simple concept matches a composed one. If the matching is possible, we branch these concepts and perform another semantic analysis to check if there is no semantic conflicts. This detection is performed after the semantic conflicts detection (concept matching) and before the detection of context heterogeneity, so that, if a context heterogeneity occurs when we detect structural conflict, it could be adapted with the CA discovery algorithm.

5.2.3 Syntactic conflicts

The syntactic conflict detection is the last task of the detection. Once the semantic and structural detection have been performed, the system checks the data syntax that will flow between services during workflow execution. If the system detects a syntactic conflict, the system creates a log of this conflict to be notified to the workflow execution program. This type of conflict is resolved in the end by the workflow execution program that performs the calls to services, and in case of syntactic heterogeneity, automatically converts data to the required format.

Once these conflicts has been tracked, the system is able to generate a mediated workflow, which allows to correctly execute service calls.

5.3 Generation of Mediated Workflows

Once the conflicts have been detected, we search for mediation services in a distributed registry (see Section 6.2). Discovered mediation services are inserted in the workflow and original links between services are removed. These mediation services are used to transform, translate and adapt data from services to others. We call a workflow with mediation services a "mediated workflow". Once the mediated workflow is ready, the workflow execution is triggered. Firstly, our execution engine identifies and stores in a list the services that are not involved as targets in any `link` element, to be invoked in the first place. Secondly, it identifies `link` elements that have as source a service from the list and collects the corresponding target elements (identifying services) in a new list. This new list feeds the next iteration of our execution. The recursion terminates when a list is generated that does not connect to any links. We give different identifiers to services that are subject to multiple invocations in the workflow so that the process terminates correctly. A service with multiple `link` dependencies is invoked in the latest step where a dependency occurs (guaranteeing data synchronization). Our algorithm generates the execution steps of a workflow while respecting the data dependencies described in the `link` elements.

6 Publication and Discovery of Mediation Services

In the context of the Web, a large number of service providers use different ontologies which leads to numerous potential semantic conflicts. In our architecture, a plethora of CA mediation services, designed and deployed by the providers of business services, provide atomic conversions from one context to another. Therefore, we rely on a distributed hash table (DHT) structure based on consistent hashing [16] to set up a distributed registry for data mediation services and allow for efficient discovery. This distributed setup opens possibilities for the emergence of complex mediation operations as combinations of different CA mediation services available in the DHT. The administration and maintenance of CA mediation services is at the charge of service providers, and becomes an incentive to promote their business services and gain market shares. We present our distributed data mediation services registry and detail the publication process hereafter.

6.1 Publication of mediation services in a distributed registry

Mediation services are published on a DHT that stores key-value pairs for distributed data items and allows, given a key, to determine the computer (node) storing the associated value using a hash function, which is also used to assign a key ID to each node and a key k to each data item. In our DHT, the nodes are mapped onto a circular identifier space called ring. Each node in the ring is responsible of the interval of keys between its key and the key of its predecessor node in the ring excluded. In this way, a data item with a key k_d is stored on the first node in the identifier space such as $k_d \leq ID_i$ where ID_i the key of that node. Our distributed registry is set up as a P2P overlay network over an existing Web services provider's network. Each provider is then mandated by a peer representing a node in our DHT, where ID keys are computed from the IP addresses of peers.

Information is distributed among the different nodes based on the semantic concepts annotating their input parameters and associated contexts. Thus, the key of a data mediation service advertised by our registry is formed by a couple (c_{in},$ctxt_{in}$) where: c_{in} is the semantic concept of the service input and $ctxt_{in}$ its associated context dimension. The "value" associated to this key is formed by a list of CA mediation services, each represented by a couple ($ctxt_{out}, URI_{desc}$) where $ctxt_{out}$ is the semantic concept of the service output and URI_{desc} is the URI of the service description. Then, CA mediation services that handle the same input data c_{in} in a same context dimension are published on the same node. Structural and syntactic mediation services also are published in the DHT and refer to specific instances in ontologies that describe data structure and syntax. For these services, the context change implies structural or syntactic changes. The SHA-1 hash function computes the keys associated to nodes and service descriptions.

Each node maintains a routing table (also called finger table) with a small number of references to other node (O log(N) references, where N is the number of nodes). When a node receives a query, the node offering the requested data will be located by routing via O log(N) hops. We use a direct storage technique to store data mediation service information in our DHT. Information about services is copied to the node responsible for them. In addition, since nodes can leave the network, each published service description is replicated to the next and previous nodes to increase its availability. The organization of service descriptions inside our DHT network is maintained based on their keys (i.e. (c_{in},$ctxt_{in}$)). Then, in order to publish a new data mediation service, we first extract these elements from its description.

Next, the value of its key is computed (using SHA-1) to locate the node in the ring wherein the data mediation service information will be published.

6.2 Discovery of Mediation Services

In a workflow, when a service S_x sends data according to a concept and a context ($S_x.c_{out}, S_x.Ctxt_{out}$) to a service S_y whose input parameter is associated to a similar concept with a different context ($S_y.c_{in}, S_y.Ctxt_{in}$), data interpretation inconsistencies and other conflicts can happen. Each inconsistency can be resolved through one or several mediation services converting and adapting the data provided by S_x to meet the target context expected by S_y. Retrieving these services, if they exist, is a relatively simple process based on our distributed registry. The discovery of a composite mediation service can be considered as a path-finding problem where the aim is to identify the smallest set of services (shortest path) to resolve a conflict (i.e. convert data as a concept instance from $S_x.Ctxt_{out}$ to $S_y.Ctxt_{in}$).

6.2.1 Mediation service matrix.

Our solution for mediation service discovery relies on a mediation service matrix to identify the optimal solution in terms of number of mediation services involved. The mediation service matrix summarizes the possibilities the DHT offers in a data structure that describes available mediation paths. Each cell contains URIs of mediation services required to convert from a context to another one. When several mediation services are required, they are stored in the cell as an ordered list. To convert a value from a context to another, we have to read the right cell in the matrix, and fetch the optimal composition of mediation services to perform the conversion.

6.2.2 Matrix generation.

The generation of the matrix relies on calls to the DHT coupled to a breadth-first search algorithm that explores available paths for each context dimension. It generates a $n \times m$ matrix $M = (a_{i,j})$ where n is the number of input dimensions available in the DHT, m is the number of output dimensions available in the DHT and $a_{i,j}$ is the smallest set of services to convert from dimension i to j.

As a first step, our algorithm calls the DHT to provide for URIs of mediation services and stores the URI of each single service in the corresponding cell, according to the context dimension the service takes as input and returns as output. As a second step, our algorithm follows the breadth-first search algorithm to find the smallest combination of services for a given path, reusing the URIs of services already stored in the matrix. The paths obtained are added to the matrix. Table 1 shows the matrix associated to the `do:price` concept, and the combinations generated by the breadth-first search algorithm (from EUR to GBP, from EUR to CNY and from JPY to GBP).

7 Experiments

In this section, we first introduce the prototype we implemented as a proof of concept of our approach and we show in details how our solution resolves data inconsistencies according to our motivating scenario. We discuss then the scalability of our mediation approach.

e.g. Mediation service matrix for the `Price` concept

	USD	JPY	CNY	GBP
EUR	uri_A	uri_B	1:uri_B 2:uri_D	1:uri_A 2:uri_C
USD				uri_C
JPY			uri_D	1:uri_D 2:uri_E
CNY				uri_E

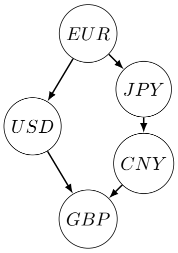


Table 1 Sample mediation service matrix and graph representation

7.1 Implementation

We deploy our python implementation as a RESTful service relying on the Apache WSGI module. We also designed a generic Javascript drag and drop GUI which acts as a client of our implementation. This interface allows user to easily design a service workflow that will be sent to our service[d]. Upon user interaction with the GUI, we generate an XML file representing the workflow relying on our simple workflow language (described in Section 5.1), ie. a graph where nodes are service (identified by an id and a description URI) and links are dependencies between these services. We also provide a field where user have to set its input data in XML format, representing input fields of the services that don't have dependancies. This workflow and the input data is then sent to the service execution engine.

The execution engine first parses the workflow and extracts service couples from the `link` elements to check that their semantic annotations match as explained in Section 5.2.1. Each service description, described in Notation3 according to the Minimal Service Model, is analyzed to extract concepts associated with it input and output parameters ($msm : messagePart$) using the python RDFlib library. With the help of domain ontologies, we retrieve the domain concepts attached to each parameter.

We exploit the RDF graph that describes the domain ontology, the conflictual aspects ontology, and service descriptions and query it, executing SPARQL queries. These queries allows us to verify wether or not the concepts involved in data exchange are "compatible", ie. the relation between them being subsumption (`rdfs:subClassOf`) or equivalence (`owl:SameAs`). This analysis initially provides the means to check the global semantic compatibility of services in the composition.

7.1.1 Conflict Detection

Semantic conflicts are detected on pairs of matching concepts, their context descriptions are matched. Conflicts are raised when context elements do not match. In our scenario, we identify the following semantic conflicts:

- phone numbers: (concept phoneNumber, context: FR, UK)
- dates and timestamps: (concept datetime, context: FR, UK)
- price: (concept: price, context: VATRate: FR, VATRate: UK, currency:EUR, currency:GBP)

[d] Prototype available: `http://soc.univ-lyon1.fr/DMaaS/Interface/index.html`.

Structural conflicts are identified when several concepts of a service match one concept of another service. We rely on the `rdf:bag` and `rdf:seq` properties to raise such conflicts. In our scenario, we identify the following structural conflicts:

- the `CompanyName` element of the `PurchaseOrder` complex output is matched to the `name` input of the `SearchCustomer` service
- the `PurchaseOrder` output is matched to the `OrderType` input of the `CreateNewOrder` service. These are two complex structures that connect to each other.

Syntactic conflicts are detected using the MSM service descriptions that make explicit the data format services use. In our scenario, we parse the MSM descriptions of services and detect when connected services use JSON and XML as data formats.

7.1.2 Conflict Resolution

On conflict detection, we search for a relevant mediation service in our distributed repository of services. We used a Distributed Hash Table, using the Python `gevent-dht`[e] package, which is a module that provides tools to create and connect nodes to a DHT network. Our Python engine creates a node and connects it to the DHT, the discovery process is then performed from this node. The chord ring stores key-values pairs representing items, allowing us to store the mediation service descriptions in a complex data structure in the DHT. Each mediation service that resolves structural, semantic and syntactic inconsistencies is stored into the DHT according to its input and output definition, i.e. $((ConceptIn, ContextIn), (ContextOut))$, as a key, and according to the URI of its service description as a value. Figure 4 shows how our solution generates a mediated workflow from our scenario.

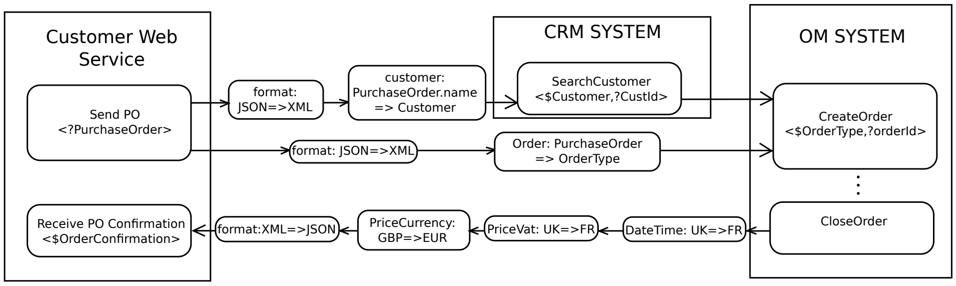

Customer Web Service
Send PO <?PurchaseOrder>
Receive PO Confirmation <$OrderConfirmation>
format: JSON=>XML
customer: PurchaseOrder.name => Customer
CRM SYSTEM
SearchCustomer <$Customer,?CustId>
OM SYSTEM
CreateOrder <$OrderType,?orderId>
CloseOrder
format: JSON=>XML
Order: PurchaseOrder => OrderType
format:XML=>JSON
PriceCurrency: GBP=>EUR
PriceVat: UK=>FR
DateTime: UK=>FR

Figure 4: Mediated Workflow Generated for our Scenario

7.2 Scalability Evaluation

In previous work [17], we investigated the scalability of our mediation algorithm against an increase in the number of DHT nodes in our distributed registry. We tested our mediation

[e] Gevent DHT module : `http://pypi.python.org/pypi/gevent_dht`

algorithm using a test collection of 1000 randomly generated service descriptions, and we gradually increase the number of nodes forming our DHT from 3 to 100. The mediation procedure in our different experiments finished in milliseconds. For instance, mediating a composite service through a distributed registry formed by 100 nodes and proposing 1000 mediation service descriptions was carried out in 294 ms. It is worth mentioning that in a real world scenario, the number of existing mediation services (resp. DHT nodes) can be hundreds (resp. tens) but should not be thousands (resp. hundreds) normally. Consequently, the performance of our mediation approach is satisfactory in real situations.

8 Related Work

In the following, we provide an overview of related work in the field. We organize this section along the following categories that relate to our contribution in this paper: mediation in service composition, and distributed discovery of services.

8.1 Mediation in service composition

The need for mediation appears when composing services in a common workflow or when trying to automate service invocation. Most research efforts around mediation can be found in the context of service composition, and have focused on protocol mediation such as in the work of [18, 19, 20]. Data mediation at the semantic level is generally envisioned as the matching of concepts at the domain ontology level, which is sometimes deemed as sufficient to solve the interpretation problems, such as in [21]. It is also required when data heterogeneity happens due to the diversity of semantic representation languages [22]. However, the work of [23, 24, 25] shows that semantic-level compatibility does not ensure compatibility on lower levels, hence in these works the authors provide mechanisms to perform low-level data conversion.

The data interpretation problem has been only addressed by a few works in the community. Most work on semantic mediation for Web services rely on one or several domain ontologies to check that some service output can be accurately related to another service input. Subsumption relationships are designed or inferred from the knowledge contained in these domain ontologies. Then, most work take for granted that the interpretation of domain ontology concepts is consistent in general [21]. The work around context [26, 27, 5] addresses the data interpretation problem with trees of semantic objects that make data interpretation explicit. Context representation based on semantic objects is interesting for data mediation but semantic objects require specific tools for their manipulation. Dietze et al. [28] rely on mediation spaces to describe the context of data as a multidimensional space where each aspect is a point on a vector. While mediation spaces offer interesting properties such as easy data conversion via Euclidian geometry, they present some drawbacks when it is needed to describe non-numerical information. The work of [6] propose an alternative solution with the use of conflictual aspect ontologies. Conflictual aspect ontologies (CAO) are specific ontologies that describe the different aspects that can be subject to mediation and provide information about data conversion from a conflictual aspect to another. In the present work, we build on this idea to enable mediation between services.

8.2 Distributed discovery of services

The problem of distributed service discovery is generally solved with federation-based approaches [29, 30, 31]. Other approaches [32, 33, 34, 35] promote the use of P2P-based solutions to overcome the management and scalability issues that can be observed while dealing with several registries. The authors in [32] present an architecture based on a semantically clustered P2P network of registry peers. Each registry peers cluster is indexed by a super peer called the index peer that stores the index information of the registry peers in a tree-based search data structure. In this architecture, a received search query will be routed by the index peers to the adequate registry peer based on the services' semantic and functional descriptions. In [33, 34], the registries' nodes are organized based on keywords extracted from the Web service descriptions. pService [35] is another P2P-based Web services discovery architecture where the registry is set up using CHORD and a data structure, called skip graph, to enhance the query routing. The authors use the service names as keys to set up their CHORD ring.

8.3 Summary

Our solution presents the following specificity with respect to the literature. First, our mediation solution promotes service oriented architecture to boost the integration of mediation components as services. Second, the mediation services to be discovered have the same functionality ("*convertTo*") and are indexed in our distributed P2P registry based on the semantic concepts annotating their input elements and associated contexts, to allow grouping in a same node all the services handling a specific kind of data, thus reducing message flow during service discovery. Third, we explore complex possibilities for conversion by connecting inputs and outputs of mediation services through the use of a mediation service matrix. Our discovery system provides composite services if no atomic ones are available. To sum up, we put together a service oriented strategy based on CAO and a distributed publication and discovery of mediation services, which makes our proposal innovative with respect to related work.

9 Conclusion

Composing Web services to achieve a certain goal, is a fundamental promise of service-oriented computing paradigm. However, the distributed nature of services hampers their composition with data heterogeneity problems that can block the execution of a composite service. In this paper, we address the data heterogeneity problems that occur during the service composition process. We propose a Decentralized Mediation-as-a-Service (DMaaS) architecture that enables the decentralized publication and discovery of mediation services, and we provide a solution to enable their automatic integration into data-driven service workflows. Our architecture enforces separation of concerns at the conceptual level with the introduction of conflictual aspect ontologies to make data interpretation explicit. We also introduce data mapping services to resolve structural and syntactic conflicts in data exchanges. Our implementation builds on a Python execution engine that has been tested over the SWS challenge scenario.

Our future research work will focus on extending our architecture so that it can handle other data concerns such as data quality or sanity check, and perform additional operations on data. We also plan to consider non-functional requirements when discovering DM services. For instance, availability and response time could be considered when defined in a service consumer's contract, execution environment requirements (such as power, storage, and memory) could be considered in case of services executed on portable devices, and so on. As part of our implementation perspectives, we plan to enhance our composition engine with advanced features such as state management and workflow verification to be able to interact with all types of services.

References

[1] Tomas Vitvar, Maciej Zaremba, Matthew Moran, and Adrian Mocan. Mediation using WSMO, WSML and WSMX. In Charles Petrie, Tiziana Margaria, Holger Lausen, and Michal Zaremba, editors, *Semantic Web Services Challenge*, volume 8 of *Semantic Web and Beyond*, pages 31–49. Springer US, 2009.

[2] Dimka Karastoyanova, Branimir Wetzstein, Tammo Van Lessen, Daniel Wutke, Jörg Nitzsche, and Frank Leymann. *Semantic Service Bus: Architecture and Implementation of a Next Generation Middleware*, pages 347–354. IEEE Computer Society, 2007.

[3] Antonio J. Roa-Valverde and José Francisco Aldana Montes. Extending esb for semantic web services understanding. In Robert Meersman, Zahir Tari, and Pilar Herrero, editors, *OTM Workshops*, volume 5333 of *Lecture Notes in Computer Science*, pages 957–964. Springer, 2008.

[4] Michael Mrissa, Mohamed Sellami, Pierre De Vettor, Djamal Benslimane, and Bruno Defude. A Decentralized Mediation-as-a-Service Architecture for Web Service Composition . In *22nd IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE 2013, Hammamet, Tunisia, June 17-20, 2013*, June 2013.

[5] Michael Mrissa, Chirine Ghedira, Djamal Benslimane, Zakaria Maamar, Florian Rosenberg, and Schahram Dustdar. A context-based mediation approach to compose semantic web services. *ACM Trans. Internet Techn.*, 8(1), 2007.

[6] Idir Amine Amarouche, Karim Benouaret, Zaia Alimazighi Djamal Benslimane, and Michael Mrissa. Context-driven and service oriented semantic mediation in daas composition. In *International Conference on Networked Digital Technologies (NDT'2012), Canadian University od Dubai, UAE.*, 2012.

[7] Edsger W. Dijkstra. *Selected writings on computing: a personal perspective*. Springer-Verlag New York, Inc., New York, NY, USA, 1982.

[8] Holger Lausen and Joel Farrell. Semantic annotations for WSDL and XML schema. W3C recommendation, W3C, August 2007. http://www.w3.org/TR/2007/REC-sawsdl-20070828/.

[9] Carlos Pedrinaci and John Domingue. Toward the next wave of services: Linked services for the web of data. *J. UCS*, 16(13):1694–1719, 2010.

[10] Mohamed Sellami, Walid Gaaloul, and Bruno Defude. A decentralized and service-based solution for data mediation: the case for data providing service compositions. *Concurrency and Computation: Practice and Experience*, pages n/a–n/a, 2013.

[11] Daniele Turi, Paolo Missier, Carole A. Goble, David De Roure, and Tom Oinn. Taverna workflows: Syntax and semantics. In *eScience*, pages 441–448. IEEE Computer Society, 2007.

[12] Edward A. Lee and Steve Neuendorffer. Moml - a modeling markup language in xml - version 0.4, 2000.

[13] Web Services Business Process Execution Language Version 2.0. Technical report, OASIS Web Services Business Process Execution Language (WSBPEL) TC, April 2007.

[14] David Martin, Mark Burstein, Erry Hobbs, Ora Lassila, Drew Mcdermott, Sheila Mcilraith, Srini Narayanan, Bijan Parsia, Terry Payne, Evren Sirin, Naveen Srinivasan, and Katia Sycara. OWL-S: Semantic Markup for Web Services. Technical report, November 2004.

[15] Tomas Vitvar, Jacek Kopecky, Jana Viskova, and Dieter Fensel. Wsmo-lite annotations for web services. In Manfred Hauswirth, Manolis Koubarakis, and Sean Bechhofer, editors, *Proceedings of the 5th European Semantic Web Conference*, LNCS, Berlin, Heidelberg, June 2008. Springer Verlag.

[16] Ion Stoica, Robert Morris, David R. Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM*, pages 149–160, 2001.

[17] Mohamed Sellami, Walid Gaaloul, and Bruno Defude. Data mapping web services for composite daas mediation. In *21st IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE 2012, Toulouse, France, June 25-27*, pages 36–41, 2012.

[18] Liliana Ardissono, Roberto Furnari, Giovanna Petrone, and Marino Segnan. Interaction protocol mediation in web service composition. *Int. J. Web Eng. Technol.*, 6(1):4–32, 2010.

[19] Yanhua Du, Xitong Li, and PengCheng Xiong. A petri net approach to mediation-aided composition of web services. *IEEE T. Automation Science and Engineering*, 9(2):429–435, 2012.

[20] Xitong Li, Yushun Fan, Stuart E. Madnick, and Quan Z. Sheng. A pattern-based approach to protocol mediation for web services composition. *Information & Software Technology*, 52(3):304–323, 2010.

[21] Shankar Ponnekanti and Armando Fox. Interoperability among independently evolving web services. In Hans-Arno Jacobsen, editor, *Middleware*, volume 3231 of *Lecture Notes in Computer Science*, pages 331–351. Springer, 2004.

[22] Tho T. Quan, Cach N. Dang, Ngan D. Le, Chattrakul Sombattheera, and Quan Vu Lam. Automatic composition and mediation on multiple-language semantic web services. In *Multi-disciplinary Trends in Artificial Intelligence - 5th International Workshop, MIWAI 2011, Hyderabad, India, December 7-9, 2011.*, pages 51–62, 2011.

[23] Bruce Spencer and Sandy Liu. Inferring data transformation rules to integrate semantic web services. In *The Semantic Web - ISWC 2004: Third International Semantic Web Conference,Hiroshima, Japan, November 7-11, 2004.*, pages 456–470, 2004.

[24] Shawn Bowers and Bertram Ludäscher. An ontology-driven framework for data transformation in scientific workflows. In Erhard Rahm, editor, *DILS*, volume 2994 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2004.

[25] Veli Bicer, Gokce Laleci, Asuman Dogac, and Yildiray Kabak. Artemis message exchange framework: semantic interoperability of exchanged messages in the healthcare domain. *SIGMOD Record*, 34(3):71–76, 2005.

[26] Cheng Hian Goh, Stéphane Bressan, Stuart E. Madnick, and Michael Siegel. Context interchange: New features and formalisms for the intelligent integration of information. *ACM Trans. Inf. Syst.*, 17(3):270–293, 1999.

[27] Xitong Li, Stuart E. Madnick, Hongwei Zhu 0002, and Yushun Fan. Reconciling semantic heterogeneity in web services composition. In Jay F. Nunamaker Jr. and Wendy L. Currie, editors, *International Conference on Information Systems, ICIS, Phoenix, Arizona, USA, December 15-18, 2009*, page 20, 2009.

[28] Stefan Dietze, Alessio Gugliotta, John Domingue, and Michael Mrissa. Mediation spaces for similarity-based semantic web services selection. *Int. J. Web Service Res.*, 8(1):1–20, 2011.

[29] Thomi Pilioura and Aphrodite Tsalgatidou. Unified publication and discovery of semantic web services. *ACM Transactions on the Web (TWEB)*, 3(3), 2009.

[30] Mohamed Sellami, Walid Gaaloul, and Samir Tata. Functionality-driven clustering of web service registries. In *IEEE International Conference on Services Computing, SCC 2010, Miami, Florida, USA*, 2010.

[31] Kunal Verma, Kaarthik Sivashanmugam, Amit Sheth, Abhijit Patil, Swapna Oundhakar, and John Miller. Meteor-s wsdi: A scalable p2p infrastructure of registries for semantic publication and discovery of web services. *Inf. Technol. and Management*, 6(1):17–39, 2005.

[32] Evren Ayorak and Ayse Basar Bener. Super peer web service discovery architecture. In *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, April 15-20, 2007, Istanbul, Turkey*, pages 1360–1364. IEEE, 2007.

[33] Cristina Schmidt and Manish Parashar. A peer-to-peer approach to web service discovery. *World Wide Web*, 7:211–229, 2004.

[34] Bin Xu and Dewei Chen. Semantic web services discovery in p2p environment. In *ICPPW '07: Proceedings of the 2007 International Conference on Parallel Processing Workshops*, page 60. IEEE Computer Society, 2007.

[35] Gang Zhou and Jianjun Yu. pservice: Towards similarity search on peer-to-peer web services discovery. In *The First International Conference on Advances in P2P Systems, AP2PS 2009, 11-16 October 2009, Sliema, Malta*, pages 111–115, 2009.