Exchanging Data Agreements in the DaaS Model

Hong-Linh Truong*, Schahram Dustdar*, Joachim Götze†, Tino Fleuren†, Paul Müller†, Salah-Eddine Tbahriti‡,
Michael Mrissa‡ and Chirine Ghedira‡
*Distributed Systems Group, Vienna University of Technology, {truong,dustdar}@infosys.tuwien.ac.at
†Department of Computer Science, University of Kaiserslautern, {j_goetze,fleuren,pmueller}@informatik.uni-kl.de
‡Université de Lyon, CNRS, Université Lyon 1, LIRIS UMR5205, {firstname.surname}@liris.cnrs.fr

***Abstract*—Rich types of data offered by data as a service (DaaS) in the cloud are typically associated with different and complex data concerns that DaaS service providers, data providers and data consumers must carefully examine and agree with before passing and utilizing data. Unlike service agreements, data agreements, reflecting conditions established on the basis of data concerns, between relevant stakeholders have got little attention. However, as data concerns are complex and contextual, given the trend of mixing data sources by automated techniques, such as data mashup, data agreements must be associated with data discovery, retrieval and utilization. Unfortunately, exchanging data agreements so far has not been automated and incorporated into service and data discovery and composition. In this paper, we analyze possible steps and propose interactions among data consumers, DaaS service providers and data providers in exchanging data agreements. Based on that, we present a novel service for composing, managing, analyzing data agreements for DaaS in cloud environments and data marketplaces.**

I. Introduction

Recent advances in devices and networks have enabled data sharing on the Internet. Individuals and companies have collected and analyzed several data sources to provide different types of data to data customers (e.g., social media data[1]). Furthermore, regulations have also been changed, introducing the so-called open data initiatives[2]. We have observed that in many cases, data is open and free, but licensed (such as in $data.gov$) as well as data is bound to commercial licenses (e.g., some data in Infochimps and Azure Data Marketplace[3]). Moreover, the Data-as-a-Service (DaaS) model [1] is increasingly employed for offering data in the cloud.

As studied in [1], data concerns are diverse, covering, e.g., data licensing, quality of data, and law enforcement. Based on these concerns, different conditions on data can be established. We call a set of such conditions applied to a single or multiple data assets *data agreement*. In some cases, data agreements might be equivalent to lawful data contracts. To date, several discussions on data concerns, in particular data licensing, have been raised in the news, forums and blogs. However, models and techniques for supporting automatic data agreements processing and exchanging in DaaS and cloud-based data marketplaces have not been in the research focus. Interestingly, some DaaS providers sell data to data consumers via pay-as-you-go or subscription APIs but to which extent sold data can be used is not clearly mentioned, specified and associated with the data. We believe that being able to dynamically and automatically determine and associate data agreements with data assets at runtime is a crucial issue. In particular, in the DaaS model, a single DaaS service provider can interact with multiple data providers to offer different types of data to different consumers, and data agreements associated with their exchanged data may be dynamically modified.

Given the lack of understanding of how DaaS service providers and data providers could work together to serve data consumers, with respect to models and protocols for data agreements, in this paper we propose techniques and services for supporting the creation, management, analysis, and association of data agreements with data in the cloud. We do not work on specifications for data agreements, but focus on models and service components that can be used by DaaS service and data providers to exchange their data agreements by exploiting different possible interactions among data services, data providers and consumers. The main contributions of our paper are:

- We propose novel models for encapsulating data agreements and for exchanging data agreements among DaaS service providers, data providers and consumers.
- We introduce a novel Data Agreement Exchanging Service (DAES) for enabling the composition, analysis and management of data agreements.

To demonstrate our work, we present several possibilities to utilize our service with real-world DaaSs.

The rest of this paper is organized as follows: Section II discusses the background and related work. Our proposed data agreement exchange metamodel, annotation techniques and interaction models for DaaS are presented in Section III. We present the design of a service for exchanging data agreements in Section IV. Section V presents our experiments. Section VI summarizes the paper and outlines our future work.

II. Background and Related Work

A. DaaS Service Provider and Data Providers

In our work, we focus on relationships among data consumers, DaaS service providers and data providers, shown in Figure 1. In our model, one DaaS service provider can interact with multiple data providers. A data provider can

[1] http://gnip.com/
[2] http://www.data.gov/opendatasites
[3] http://www.infochimps.org/ and https://datamarket.azure.com/

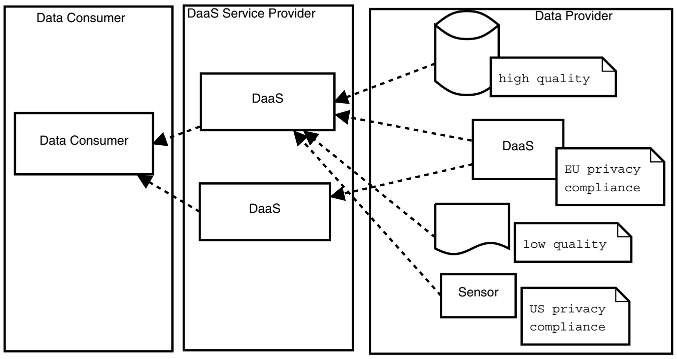

Figure 1. Interactions in the DaaS model

pass data to a DaaS on-demand, based on requests from the DaaS. A data consumer can use a pay-as-you-go model to request data from a DaaS service, which, in turn, searches different data providers, if they can provide some types of data in order to fulfill the request. The data providers can set different constraints on the data assets they provide. In the DaaS model, not only the interactions between a DaaS service provider and a data provider are very dynamic, but the issues of data agreement management are complex as well due to the rich types of data assets to be exchanged.

We see that on-demand data provisioning in the DaaS model introduces new problems related to data availability and agreement. In order to satisfy their consumers' needs, DaaSs may search for new data assets which can be provided, but in many cases these data assets come with different agreements from different providers. Therefore, there is a need to provide a generic interaction model that enables DaaS and data providers to exchange agreements for data (e.g., during the negotiation between DaaS service providers and data providers). There is also a need, at the DaaS level, to make a distinction between the different types of agreements, and to be able to sort out which data assets they apply to. In our work we will focus on a model/service for DaaS providers, data providers and data customers to exchange data agreements in pay-as-you-go and subscription data access.

B. Data Agreement Specifications

In most cases, data are accompanied with data agreements which are written for humans, such as described in Open Data Commons[4]. One of the most popular forms of data agreements is data licensing. ODRL [2] is a specification that can be used to specify data agreements but it is not designed for data on the Web. ONIX-PL is another effort to develop licenses for digital resources[5]. Some work incorporate some data terms in XML but there exists no standard or widely accepted way to do this. We expect that data agreements will be specified in machine-readable forms, although we see the lack of specifications for data agreements. In our view, in the DaaS model and data marketplaces, there would exist different types of data agreements that should be exchanged. Therefore, in this paper, we focus on managing and exchanging data agreements that can integrate different relevant specifications.

C. Exchange of Data Agreements

License management in Grid and Cloud computing such as [3] does not support data agreement interactions in the DaaS model, which is much more flexible. Our previous work dealt with licensing for data in the Grid [4]. However, as we discuss later on, data agreements are much more broader and licenses are just a part or a type of agreements. Due to its difference, Grid licensing cannot be applied to DaaS data agreements. Digital right management techniques [5] are also relevant to data agreements, however, existing techniques [6], [7] are not designed to work in dynamic DaaS interactions with diverse types of structured and unstructured data in cloud-based DaaS and open data marketplaces, such in Infochimps and Azure Data Marketplace. Our work focuses on data agreement exchange to support data discovery and compliance, as a part of digital right managements in the DaaS model.

Researchers have also focused on data policing. For example, Speiser and Studer propose a self-policing policy language in [8] and use a policy engine to apply policies during composition. Different from them, we allow different agreements to be defined and managed so that any interesting parties can obtain the agreements and perform data policing.

III. Interactions in Data Agreement Exchange

A. Metamodel for Data Agreements

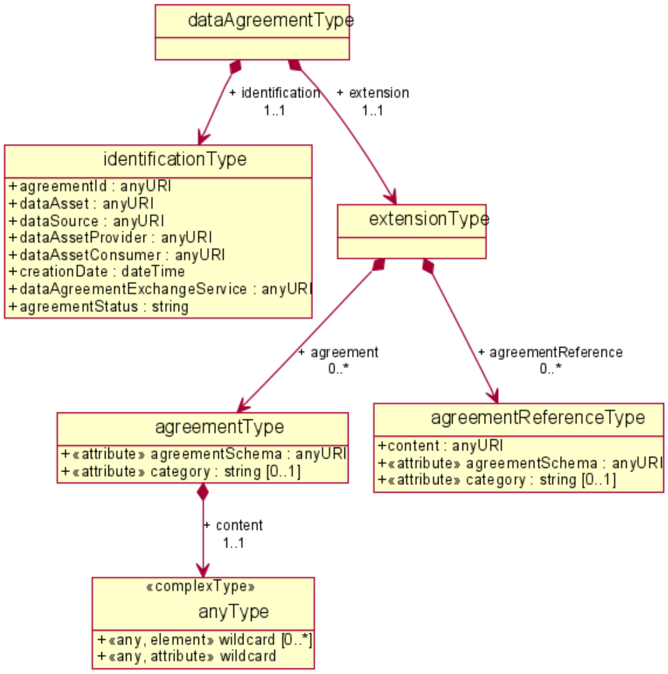
dataAgreementType
+ identification
1..1
+ extension
1..1
identificationType
+agreementId : anyURI
+dataAsset : anyURI
+dataSource : anyURI
+dataAssetProvider : anyURI
+dataAssetConsumer : anyURI
+creationDate : dateTime
+dataAgreementExchangeService : anyURI
+agreementStatus : string
extensionType
+ agreement
0..*
+ agreementReference
0..*
agreementType
+«attribute» agreementSchema : anyURI
+«attribute» category : string [0..1]
agreementReferenceType
+content : anyURI
+«attribute» agreementSchema : anyURI
+«attribute» category : string [0..1]
+ content
1..1
«complexType»
anyType
+«any, element» wildcard [0..*]
+«any, attribute» wildcard

Figure 2. Structure of the meta model for encapsulating data agreements

Data agreements can be classified into different categories, such as, agreements on data licensing, on data privacy compliance, or on quality of data. To support data agreements

[4] http://www.opendatacommons.org
[5] http://www.editeur.org/21/ONIX-PL/

exchange, we need to encapsulate different categories, contents and other metadata about agreements. In doing so we develop a metamodel to capture information about data agreements.

Figure 2 shows our simplified metamodel that can be used among data consumers, DaaS service providers, and data providers in order to exchange agreements. The metamodel consists of a common part (indicated by `identificationType`) and an extension part (identified by `extensionType`). The common part contains all managing information for identifying the data asset to which the agreement applies (element `dataAsset`), possible original data source (element `dataSource`), the data asset provider (element `dataAssetProvider`) which provides the data[6], and the data asset consumer (element `dataAssetConsumer`). The common part also stores the service used to support the agreement exchange among data asset consumers and providers. This service, specified by `dataAgreementExchangeService`, plays the role of a mediator that stores and manages data agreements as well as and performs various features atop agreements. Furthermore, the common part stores information about the status of the agreement, e.g., `CREATED` or `AGREED`. Note that the common part does not contain any specific description of a data agreement. Numerous forms of such descriptions may exist, thus we integrate a specific description into the extension part, which can contain descriptions of several data agreement constraints. In addition, the extension part may hold the same data agreement terms described by several description languages. For example, on the case of licensed content, the same license may be described in different languages, so that different data consumers may select the most suitable language. This approach allows for a more generic and extensible model.

The extension part stores a list of data agreement constraints and capabilities (via embedded content specified by the element `agreementType` or via reference content specified by `agreementReferenceType`), which specialize to different types like quality of data, privacy, or license (indicated by `category`). Any suitable description language can be used in a data agreement section, e.g., Open Digital Rights Language (ODRL) [2]. Typical information stored in such a section could be, for example, provider, consumer, contract expiration date, data rights, compliance, and law enforcement. Multiple extension sections can be used to contain different agreement categories. Multiple agreement sections can be considered as a means of convenience and allow us to support new description languages or requirements on data agreements. Nonetheless, each data agreement section has to be complete and valid so that it can be exchanged and associated with data assets to which the agreement applies.

B. Enriching and Associating Data Assets with Agreement Terms

Data agreement can be tightly or loosely coupled to data as a means to make agreement terms that apply explicitly to data usage. Tight coupling implies exploiting the existing structure of data assets, e.g., by devising new data structures that contain agreement terms. On the contrary, loose coupling implies implementing agreements as entities that are physically distinct from data assets. On the technical level, different possibilities are available for attaching data agreements to data assets. We have identified four solutions and we discuss their advantages and drawbacks in the following:

(a) directly inserting agreements into data assets,
(b) providing two-step access to agreements and data assets,
(c) linking data agreements to the description of DaaS, and
(d) linking data agreements to the messages sent by DaaS.

Solution (a), which involves tight coupling, is particularly attractive with unstructured data, where agreement terms can be added, for example into a zip file.Tight coupling presents a general drawback: injecting and extracting agreement terms into and from data assets is a costly task. However, such a solution is quite interesting when the goal of the DaaS service provider is to tightly attach agreement to data and to enforce data usage to agreement terms in any situation. Furthermore, this solution is not scalable, as the multiplication of concerns would generate a complex interface.

With solution (b), the consumer is first given a link to data agreements, which must be accepted before data access is granted. This solution can be implemented using a similar mechanism described in OAuth[7] in which DaaS service providers can ask data providers to grant access token to data consumers when the consumers interact with the DaaS service. However, this solution might require manual interactions. This solution is easy to implement and can be reused over different data assets after deployment.

Solution (c) raises a new problem: it is not possible to distinguish which data, from which data source, is concerned with which agreement, if there are several agreements attached to the same service description. Solution (c) should then be avoided. Solution (d) presents the same drawbacks as solution (a) but is restricted to adding a URL to the service message, e.g., in XML, which is not a big change. Another possibility is to use message headers of underlying protocols (i.e., HTTP) to incorporate a link to agreement terms, for example using Web linking as described in the RFC 5988 [8].

Overall, different possibilities for associating agreement terms with data assets have different advantages and drawbacks with respect to their applicability to unstructured and structured data. Table I summarizes the result of our discussions. Based on that, we recommend that XML-based services should rely on HTTP Web linking to provide the link to agreement information to data assets provided. Accordingly, we deem appropriate for services that provide unstructured data to inject agreement terms into data assets.

C. Interaction Models for Data Agreement Exchange

Based on the meta model for data agreements and techniques to associate data assets with data agreement infor-

[6] In this case, it can be a DaaS service provider or a data provider

[7] http://oauth.org

[8] http://tools.ietf.org/html/rfc5988

coupling	tight-coupling	loose-coupling		
technical solution	(a)	(b)	(c)	(d)
structured data	(+)ciphering possible (-)requires specific client (-)not scalable (-)modifies data structure	(+)data-independent (-)manual access only	(-)service-specific data agreement (-)no enforcement possible	(+)message-specific data agreement (+)ciphering possible
unstructured data	(+)data agreement enforcement possible (+)ciphering possible (-)requires specific consumer (-)costly data agreement injection	(+) data-independent (-)manual access only	(-)service-specific data agreements (-)no enforcement possible	(+)message-specific data agreement (+)ciphering possible

Table I
SUMMARY OF ATTACHING DATA AGREEMENTS TO DATA ASSETS

mation, we devise interactions for data agreement-enriched DaaS. We consider two different situations: (i) a DaaS service provider knows its data sources and the data agreements for data associated with these sources in advance, and (ii) a DaaS service provider might or might not know its data sources in advance, and the data agreements associated with its data sources are determined at runtime. In both cases, data sources of a DaaS can be internal data sources (belonging to the DaaS) and external data sources (from other data providers).

Figure 3 presents our interaction models for exchanging data agreements in the DaaS model by utilizing a data agreement exchange as a service (DAES). When a DaaS receives a data request from a data consumer, sent in `Step (1) data request`, there are different possibilities about possible data agreements for requested data, with respect to data agreement:

- if the data sources for the request are known (e.g., in a series of transactions) and the data agreements of the requested data are known: in this case, a DaaS invokes DAES to check existing data agreements and/or to receive the data agreement content or the reference to the agreement to be applied to the requested data (`Step 2.1`). After that the DaaS will request the data from corresponding data providers (`Step 2.2`). The data providers can recheck agreements (`Step 2.3`) before sending the data back to the DaaS.
- if data agreements for data sources are not known, there are two possibilities. First, a DaaS asks data providers about data and corresponding data agreements (`Step 3.1`); the data providers offer data agreements (`Step 3.2`); the DaaS checks the offering data agreements (`Step 3.3`) before the DaaS accepts and receives the data. Second, a DaaS may invoke DAES to create new data agreements (`Step 4.1`) and then asks data providers to agree to provide data based on its proposed agreements (`Step 4.2`); the data providers check the agreements (`Step 4.3`) before sending the data to the DaaS[9].
- a DaaS can compose new data agreements for the data that it will offer to the data consumers (`Step 5`). After the data providers check the agreements (either by themselves or using DAES), the data providers return the requested data to the DaaS which, in turn, returns the data and agreements back to the data consumers. The final delivery of data assets and agreements can be in different forms, as discussed in Section III-B, e.g., agreements are packed with requested data or agreements are delivered separate from requested data.

Based on our proposed interaction models, DaaS service providers and data providers can implement both tight-coupling and loose-coupling solutions discussed in Table I. Moreover, we see the need to develop DAES's features to support the creation and validation of data agreements for different DaaS. As agreements can be stored and retrieved before, during and after data requesting, DAES could support the composition and compatibility checking of data agreements and agreement negotiation.

IV. DATA AGREEMENT EXCHANGE AS A SERVICE

Based on our models in Section III, we develop a data agreement exchange as a service (DAES) to support the management and exchange of data agreements. DAES aims at being a cloud service for data marketplaces in which different data agreement specifications can be registered, multiple DaaS providers, data providers and data consumers can use DAES to exchange their data agreements, and several agreement-specific operations, such as creation and validation, composition, and compatibility analysis, can be supported.

Figure 4 presents an overview of our DAES. At the bottom level, DAES stores different information, including agreement specifications, templates and agreement instances (concrete agreements created based on specific specifications), and DAES consumers (DaaS service providers, data providers, and data consumers). At the middle level, DAES offers two types of functionalities. First, DAES offers several agreement-specific features in which each feature will be supported by plug-ins in terms of *Data Agreement Apps*. Examples of such features are data agreement creation and validation, data agreement management, data agreement composition, and data agreement compatibility analysis (see Table II). Note that by agreement-specific, we mean that these features are differently implemented for and applied to different agreement specifications. The main reason for using plug-in models is that each agreement specification requires different mechanisms to handle its agreement instances. For example, there will be several plugins for verification of agreement compatibility (e.g., implemented based on existing algorithms). Note that the

[9] Note that data agreement negotiation can be performed. However, it is a complex issue and out of the scope of the paper.

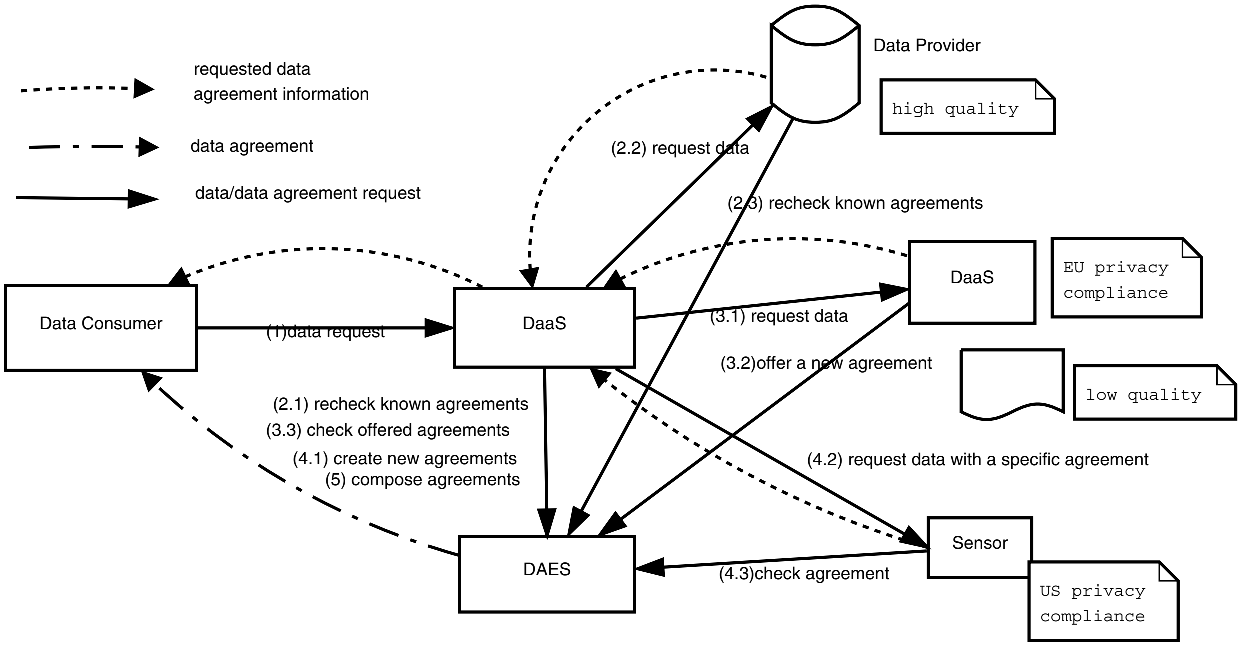
requested data agreement information
data agreement
data/data agreement request
Data Provider
high quality
(2.2) request data
(2.3) recheck known agreements
DaaS
EU privacy compliance
Data Consumer
(1)data request
DaaS
(3.1) request data
(3.2)offer a new agreement
low quality
(2.1) recheck known agreements
(3.3) check offered agreements
(4.1) create new agreements
(5) compose agreements
(4.2) request data with a specific agreement
Sensor
DAES
(4.3)check agreement
US privacy compliance

Figure 3. Possible interaction models for data enriched with data agreements

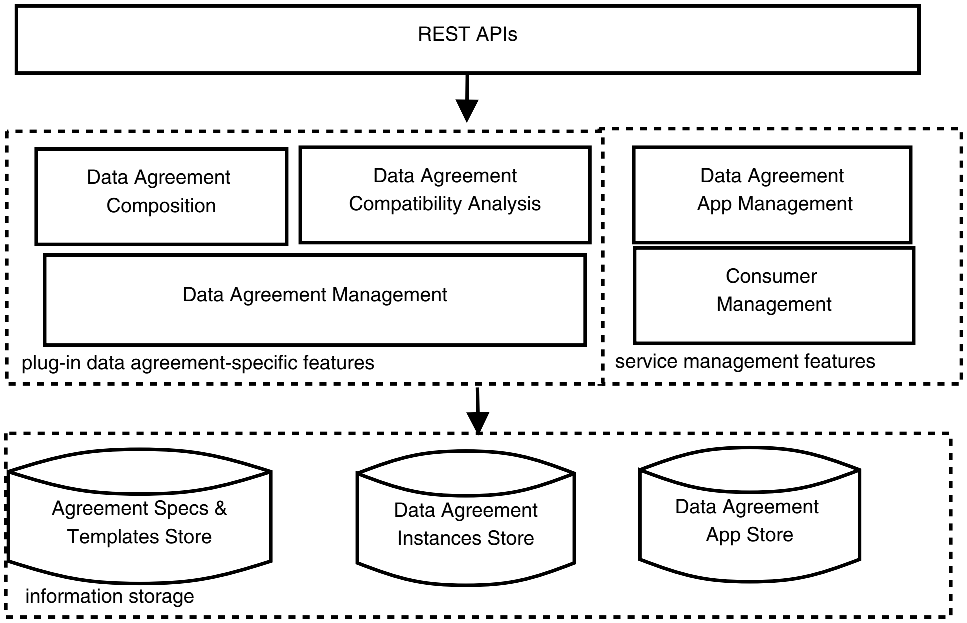
REST APIs
Data Agreement Composition
Data Agreement Compatibility Analysis
Data Agreement Management
plug-in data agreement-specific features
Data Agreement App Management
Consumer Management
service management features
Agreement Specs & Templates Store
Data Agreement Instances Store
Data Agreement App Store
information storage

Figure 4. Overview of Data Agreement Exchange as a Service (DAES)

development of *Data Agreement Apps*, e.g., for compatibility algorithms, is out of this paper.

Second, DAES offers service management features to manage agreement-specific applications and consumers. All DAES features are exposed via a set of RESTful APIs. Figure 5 describes the (simplified) information model inside DAES. We manage agreement specifications (`DataAgreementSpecification`), data agreement template (`DataAgreementTemplate`) and data agreement instances (`DataAgreementInstance`). Data agreement instances, templates and specifications are interlinked. The `AgreementContent` will be used to store concrete data agreement contents. All instances, templates and specifications will have certain information in common, specified via `AbstractDataAgreementInfo`, such as unique URI, status, provenance information, and compliant law enforcements. Given agreement instances, templates and specifications, we manage possible data agreement applications (`DataAgreementApp`) and consumers (`DAESConsumer`) that are associated with them.

Note that certain information in the above-mentioned entities can also be available in the metadata mentioned in Section

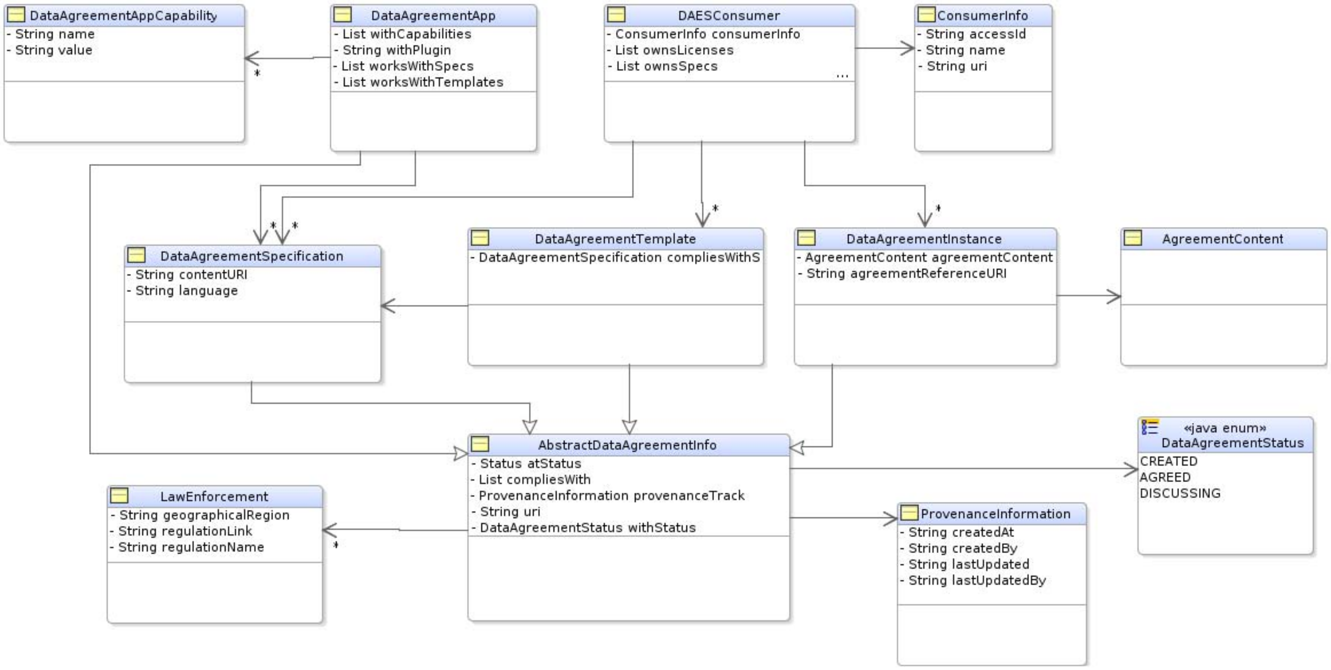

Figure 5. Main information stored in DAES

Features	Description
agreement creation	create an agreement based on DAES consumer input parameters. The agreement is created based on a specific specification.
agreement validation	check if an agreement is syntactically correct.
agreement composition	perform agreement merging, nesting, etc.
agreement compatibility analysis	check if a set of agreements are compatible
agreement management	store, retrieve, delete, and update agreements and their specifications and templates.

Table II
POSSIBLE AGREEMENT-SPECIFIC APPLICATIONS

III-A, such as specification language, law enforcement, or consumer identifiers. Here we must distinguish two different ways of utilizing metadata about data agreements. First, within DAES, metadata is used to support the search, management, and retrieval of data agreements as well as to create data agreement instances. Second, a data agreement content, stored in `AgreementContent`, is a self-contained information that must specify enough information for any consumer to interpret it and it can be shifted with data assets out of the scope of DAES, e.g., without using references to data agreements.

Our prototype of DAES is based on Jersey – and implementation of JAX-RS for RESTful Web Services – and JDeveloper. We have tested our prototype with Weblogic 10.3. In our implementation, we have intensively used URIs to identify data agreement applications, consumers, data agreement specifications, data agreement templates, and data agreement instances. Therefore, consumers can straightforwardly obtain many types of information by simply using URIs.

V. EXPERIMENTS

In this section, we illustrate examples of how to use our DAES to support data agreements in the DaaS[10].

A. *Publishing and Annotating Data Agreements*

Currently, as discussed in Section II, existing DaaS have not published data agreements that support automatic discovery data assets by using data concern terms. One popular way in current DaaS is to provide pay-as-you-go APIs in which users pay the money and get the data. However, the data is not associated with data agreements. In this example, we consider that the data returned by pay-as-you-go APIs from DaaS can be annotated with data agreements. To this end, a DaaS can build data agreements based on its consumer's subscription models and store the agreements into DAES. When an API is invoked, before returning the data to its consumer, the DaaS annotates a metadata about corresponding data agreements into the data. Listing 1 shows an example of agreement-enriched people data returned from the Infochimps People Search APIs[11].

To utilize our DAES for describing agreements associated with data to be provided by DaaS, a similar way can be applied for metadata about DaaS. For example, in cases a DaaS uses OData[12] to specify and publish its data resources using URI and HTTP-based protocols, data agreements can be annotated into the service metadata document part of OData.

[10] We provide some supplement materials at http://www.infosys.tuwien.ac.at/prototype/SOD1/daes

[11] http://www.infochimps.com/datasets/twitter-people-search

[12] http://www.odata.org

```
<?xml version="1.0" encoding="UTF-8"?>
<ns0:dataAgreement xmlns:ns0="urn:de:icsy:dataagreement" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <identification>
      <agreementId>urn:de:icsyagreement:1</agreementId>
      <dataAsset>urn:de:icsy:asset:wcs:1</dataAsset>
      <dataSource>http://gdi-arc1.gridlab.uni-kl.de/arcgis01/services/Hamburg_WFS/MapServer/WCSServer?service=WCS</
           dataSource>
      <dataAssetProvider>http://gdi-arc1.gridlab.uni-kl.de/arcgis01/services/Hamburg_WFS/MapServer/WCSServer?service=WCS</
           dataAssetProvider>
      <dataAssetConsumer>daas-test1</dataAssetConsumer>
      <creationDate>2011-07-04T20:11:15.029Z</creationDate>
      <dataAgreementExchangeService> http://sod.infosys.tuwien.ac.at:7101/services/jersey/DAES</
           dataAgreementExchangeService>
      <agreementStatus>AGREED</agreementStatus>
   </identification>
   <extension>
      <agreementReference agreementSchema="urn:de:icsy:license:wcs" category="licensing">
         <content>http://sod.infosys.tuwien.ac.at:7101/services/jersey/DAES/da/references/retrieve/license_wcs</content>
      </agreementReference>
   </extension>
</ns0:dataAgreement>
```

Listing 2. Example of metadata about a data agreement

```
{"results":[
dataagreement:{
     ...
     extension: {
        agreementReference: {
             category:''licensing'',
             agreementSchema  :''urn:at:act:tuwien:infosys:
                  license:twitter'',
             content:''http://.../DAES/da/references/
                  retrieve/peoplesearch-license'',
        }
     }
},
{"description":"Student: spatial planning, music. Location:
      Vienna. Interests: Everything
 nerdy.","location":"Vienna,
 Austria","time_zone":"Vienna","user_id":REMOVED,"
      utc_offset":"3600","name":"REMOVED","scraped_at"
      :1259592694000,"screen_name":"REMOVED"},
...}}
```

Listing 1. Metadata about an data agreement instance

B. Agreements for Geospatial Data

Let us consider an example of noise emission simulation in urban environments using geospatial data, which can only be used in compliance with specific data agreements. Typically, for such a simulation several different kinds of geospatial data are retrieved from data sources via Web services that comply with the Open Geospatial Consortium (OGC[13]) standard. Our example used two types of geospatial data:

1) Vector data from the data supplier via Web-Feature-Services (WFS): Geographical features are user defined geographic pieces of interest including, for example, streets, sewer lines and accidents, represented in shapefile format [9].
2) Terrain elevation data via Web-Coverage-Services (WCS): 2.5D raster data/ digital elevation models (DEM) coverages are objects inside a geographical area; examples of formats supported by a WCS are DTED, GeoTIFF, or NITF [10].

The two services WCS and WFS can be considered a domain-specific DaaS providing geospatial data. One of the main service operations of WCS is `getCoverage`. To demonstrate how WCS (or the WFS respectively) can utilize our DAES, we selected ODRL as a data agreement specification for WCS. Let us assume the DaaS service `daas-test1` as a consumer of the data provider WCS. WCS creates a data agreement `license_wcs`, in advance, and stored it into DAES. In our test, `license_wcs` contains data license agreement terms. It describes three ODRL assets, $ASSET1, ASSET2, ASSET3$. $ASSET1$ and $ASSET3$ can be displayed and printed as often as desired, and $ASSET2$ can be displayed, but printed only 50 times. By extracting `license_wcs` and utilizing information provided by WCS when storing `license_wcs`, DAES will store common information about the data source and the data asset provider.

When a data consumer retrieves the data agreement from DAES, DAES will return all relevant agreement information and present to the data consumer the agreement either in complete form (all data is in the agreement) or in reference form (only metadata is returned). An example of metadata is given in Listing 2. In this case, the real agreement content will be stored in DAES and can be retrieved using the agreement content URI.

C. Data Agreement App for Policy Compliance

Let us consider a mashup service that delivers Twitter posts is combined with a map service. It is required that the mashup service must enforce compatibility of data privacy policies before executing mashups. To this end, DAES can be utilized to validate policies and to check their compatibility. Let us consider a simple privacy policy described as a RDF node in which a privacy policy should be interpreted as follows: the recipient r can access the target resource dr for a purpose p in the scope of s provided that the conditions $cond$ are met and the data rights dr will be respected; the mashup service will apply one of the operations in op on the data item dr before releasing it. In order to make the compatibility check-

[13] http://www.opengeospatial.org

ing available as a DataAgreementApp in our DAES, the compatibility analysis code is wrapped into a plug-in using a generic interface for DataAgreementApp, shown in Listing 3. The interface mainly requires the development of functions setDataAgreements, setDescription, execute and getResult. Using reflection mechanisms, DAES will load corresponding DataAgreementApp, pass parameters to the app using setDataAgreements, before executing the app and receiving the result. Furthermore, information about this application will be available for searching, such as in Listing 4.

```
public class TwitterCompatibilityApp implements
     DataAgreementAppInterface {
    //...
    public String getResult() {
        return output;
    }
    public void setDataAgreements(List dataAgreements,
         boolean reference) {
        this.dataAgreements = dataAgreements;
        //....
    }
    public void execute() {
        ModelManager mm = new ModelManager();
        String agreementReference1 = (String)dataAgreements
             .get(0);
        String agreementReference2 = (String)dataAgreements
             .get(1);
        OntModel m1 = mm.loadFromAgreementContent(
             agreementReference1);
        OntModel m2 = mm.loadFromAgreementContent(
             agreementReference2);
        boolean valid1= mm.validateModel(m1);
        boolean valid2= mm.validateModel(m2);
        boolean compatible=mm.isCompatible(m1, m2);
        output="<?xml version=\"1.0\" encoding=\"UTF-8\"
             ?>\n" +
        //...
    }
    public void setDescription(DataAgreementApp description
         ) {
    //...
```

Listing 3. Example of writing a DataAgreementApp

```
<dataAgreementApp>
  <withPlugin>RDF-Policy-Twitter-Mashup-CompApp</withPlugin
       >
  <withCapabilities>
    <dataAgreementAppCapability name="compatibility" value=
         "true"/>
    <dataAgreementAppCapability name="validation" value="
         true"/>
    <worksWithSpects>
      <dataAgreementSpecification>
        <contentURI>http://sod.infosys.tuwien.ac.at:7101/
             services/jersey/DAES/daspecs/retrieve/
             twitterpolicy</contentURI>
        <language>RDF</language>
      </dataAgreementSpecification>
    </worksWithSpects>
  </withCapabilities>
</dataAgreementApp>
```

Listing 4. Example of DataAgreementApp description

VI. Conclusions and Future Work

There are dynamic interactions among data consumers, DaaS service providers and data providers in providing and consuming data assets in data marketplaces in the cloud. In such interactions, supporting the exchange and management of data agreements associated with data assets is paramount of importance. In this paper, we analyze different solutions and develop a novel service for exchanging data agreements in the DaaS model. Our service provides fundamental functionalities that can be used for further advanced solutions in cloud-based data marketplaces, such as data agreement enforcement and negotiation. To our best knowledge this is the first work that aims at proposing a service for exchanging data agreements in the DaaS model.

In our future work, we will enhance our prototype to support more interaction protocols and will integrate different data agreement applications, in particular, for agreement evaluation and composition. Furthermore, we consider to integrate our service into data market places.

Acknowledgment

This work is partially supported by the Vienna Science and Technology Fund (WWTF), project ICT08-032.

References

[1] H. L. Truong and S. Dustdar, "On analyzing and specifying concerns for data as a service," in *APSCC*, M. Kirchberg, P. C. K. Hung, B. Carminati, C.-H. Chi, R. Kanagasabai, E. D. Valle, K.-C. Lan, and L.-J. Chen, Eds. IEEE, 2009, pp. 87–94.

[2] R. Iannella, "Open digital rights language (odrl) version 1.1," World Wide Web Consortium (W3C), 2002. [Online]. Available: http://www.w3.org/TR/odrl/

[3] Y. Raekow, C. Simmendinger, P. Grabowski, and D. Jenz, "License management in grid and cloud computing," in *3PGCIC*, F. Xhafa, L. Barolli, H. Nishino, and M. Aleksy, Eds. IEEE Computer Society, 2010, pp. 9–15.

[4] J. Goetze, T. Fleuren, P. Mueller, and S. Schwantzer, "License4grid: Adopting drm for licensed content in grid environments," *European Conference on Web Services*, vol. 0, pp. 19–26, 2010.

[5] Q. Liu, R. Safavi-Naini, and N. P. Sheppard, "Digital rights management for content distribution," in *Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003 - Volume 21*, ser. ACSW Frontiers '03. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2003, pp. 49–58. [Online]. Available: http://portal.acm.org/citation.cfm?id=827987.827994

[6] W. Jonker and J.-P. Linnartz, "Digital rights management in consumer electronics products," *Signal Processing Magazine, IEEE*, vol. 21, no. 2, pp. 82 – 91, mar 2004.

[7] B.-N. Park, J.-W. Kim, and W. Lee, "Precept: a privacy-enhancing license management protocol for digital rights management," in *Advanced Information Networking and Applications, 2004. AINA 2004. 18th International Conference on*, vol. 1, 2004, pp. 574 – 579 Vol.1.

[8] S. Speiser and R. Studer, "A self-policing policy language," in *International Semantic Web Conference (1)*, ser. Lecture Notes in Computer Science, P. F. Patel-Schneider, Y. Pan, P. Hitzler, P. Mika, L. Z. 0007, J. Z. Pan, I. Horrocks, and B. Glimm, Eds., vol. 6496. Springer, 2010, pp. 730–746.

[9] P. A. Vretanos, "Web feature service implementation specification." Open Geospatial Consortium, 2005. [Online]. Available: http://portal.opengeospatial.org/files/?artifact_id=8339

[10] A. Whiteside and J. Evans, "Web coverage service (wcs) implementation specification." Open Geospatial Consortium, 2006. [Online]. Available: https://portal.opengeospatial.org/files/?artifact_id=18153