On Evaluating and Publishing Data Concerns for Data as a Service

Hong-Linh Truong and Schahram Dustdar
Distributed Systems Group, Vienna University of Technology
Email: {truong,dustdar}@infosys.tuwien.ac.at

Abstract—**The proliferation of Data as a Service (DaaS) available on the Internet and offered by cloud service providers indicates an increasing trend in providing data under Web services in e-science and business domains. While data usage and selection are dependent on different constraints established on the basis of several data concerns, for example, quality of data and data privacy, existing data service engineering approaches lack techniques to allow the evaluation, association and publishing of such concerns with data provided via DaaS. Furthermore, data sources behind DaaSs are not static but dynamically changing, thus requiring the evaluation and publishing of data concerns to be dynamic and on-the-fly as well. In this paper, we present a novel data concern-aware service engineering process for evaluating and publishing data concerns inside DaaS that covers different evaluation and publishing scopes, modes, and integration models. Based on our process, we present a framework and its implementation for the evaluation and publishing of quality of data metrics associated with data provided by DaaSs. In this paper, we also perform several experiments to demonstrate the usefulness of our framework.**

I. Introduction

Recently, there is an increasing availability of data sources accessible from the Internet. Such data sources exist in different domains, such as environmental data[1], biology data[2], statistics data[3], and social network data[4]. Such data can be static, without or with little change, e.g., company credit balances and statistic data, and be very dynamic, e.g., environmental sensor data and social network data. To further allow data consumers to access the data in an easy and interoperable way, data sources are increasingly wrapped into Web services, followed the so-called Data as a Service (DaaS) model [1], [2] and typically implemented using Web services technologies based on SOAP/REST APIs.

While the Internet and SOA technologies are drivers for the success of the data publishing and integration under the DaaS model, several DaaSs exist without or with very little information about data concerns associated with the data. Besides the data and its schema, however, using data always is constrained by several data concerns, such as quality of data, data privacy, data lifecycle and data licensing, as shown in several research [2], [3]. Due to the lack of explicit publishing data concerns, data consumers face several problems in understanding constraints applied to the data. For example, is the data *free* so that it can be composed with other sources or is the data in a *good quality* so that it does not impact on the relevance of data compositions? Such questions are quite challenging for data consumers when using the data directly and for data integrators when filtering and composing data from multiple DaaSs. DaaS service and data providers need to evaluate and provide data concerns in order to tackle such questions. Without explicit information about data concerns, data consumers also face information overloading as irrelevant results can be obtained. While several tools exist to support the development and publishing of data under the DaaS model, currently there is the lack of techniques and tools to deal with the evaluation and publishing of data concerns for DaaS.

Consider the importance of the evaluation and publishing of data concerns together with the actual offered data in DaaS. In our previous work we have performed a detailed analysis of data concerns [2]. However, the process in which data concerns are gathered, measured and published is still missing. In particular, data concerns will change as long as the data changes or new knowledge about the data is obtained. Therefore, such data concerns have to be also evaluated, managed and published on-the-fly. In this paper, we address the above-mentioned issues by contributing (i) a novel, generic data concern-aware service engineering process for DaaS and (ii) a framework for evaluating and publishing quality of data metrics for DaaS, as an implementation of our proposed process. Our process and our framework cover different evaluation scopes, modes and integration models for data concerns. This paper also presents several experiments to illustrate the usefulness of our process and framework.

The rest of this paper is organized as follows: Section II presents the background and our approach. We introduce our data concern-aware service engineering process for DaaS in Section III. Section IV describes a framework for evaluating and publishing quality of data metrics for REST-based DaaSs. We present several experiments in Section V. Related work is presented in Section VI. We conclude the paper and outline our future work in Section VII.

II. Background and Approach

We consider cases in which a developer needs to expose data sources through DaaS. In such cases, the developer will need to select data to be exposed, writing code to allow data to be retrieved and updated via service APIs, such as based

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

[1]http://www.epa.gov/tri/tridata/current_data/index.html
[2]http://www.ensembl.org/index.html
[3]http://data.worldbank.org
[4]http://docs.gnip.com/Introduction-to-Gnip

on SOAP and REST. Our main question is how to support the developer to assure that data concerns will be associated with data and service APIs so that not only the DaaS consumer can access and update data based on data concerns but the data selection and composition can also utilize data concerns.

While generally we consider a DaaS as a Web service, the DaaS model has some distinguished features from normal computational service or software-as-a-service. In our view, a DaaS can only provide data based on existing data sources or can allow any data provider to create, retrieve, update and delete her data [2]. Therefore, the *DaaS service provider* is not necessarily the same as the *DaaS data provider*[5].

- *the DaaS service provider is also the data provider*: in this case, a data provider basically wants to publish its data through its own DaaS. One example is to write DaaS using contemporary Web service development toolkits to access data via JDBC.
- *the DaaS service provider is not the data provider*: in this case, the data provider utilizes common DaaSs to publish the data. The DaaS service provider and data provider can be loosely or tightly coupled in providing data. For example, a data provider can use the Infochimps[6] to publish her data by using a pre-defined interface offered by the Infochimps. But a data provider can develop her own DaaS interfaces for her data based on a tightly coupling relationship with a DaaS service provider offering an infrastructure DaaS based on the WSO2 Data Service[7].

When engineering DaaS, we consider, on one side, the DaaS provider and data provider, which collects data and publishes the collected data based on their constraints, and, on the other side, the data service consumer, which accesses published data based on its requirements.

In our work, we assume that a set of data items is represented as a *data resource*, which can be a structured relation, like in a traditional relational database, a semi-structured relation/file, like in XML/RDF data, or an unstructured/specified data form, like images and zip files. The mapping between data items and their data resource is considered as system-specific implementation. A service s will provide APIs for accessing a set of data resources. In doing so, the implementation of s will offer a set of service operations, each will take a data request and return a data resource met the request. In the literature, several techniques have been introduced for specifying and implementing the mapping among data items, data resources, and service operations. For example, a service operation can accept a SQL statement and return data resources as a result of querying the statement on a SQL-based data source.

A data resource can be associated with several data concerns [2]. Obviously, data concerns of a data resource are related to data concerns of its data items. However, in this paper, we will not consider data at the data item level, but we focus on data resources. Data concerns can be represented by a set of metrics. When a data resource is provided under the DaaS model, its data concern metrics should be also evaluated and published. The question in this paper is how to establish a methodology that facilitates the integration of the evaluation of data concerns into the publishing of data concerns in the DaaS model. However, we will not address concrete data concern metrics that should be evaluated as well as their definitions and tools to evaluate them.

III. Data Concern-aware Service Engineering Process

In order to ensure that data concerns associated with provided data to be available and searchable to the DaaS consumer, the DaaS and data providers have to utilize several components and to conduct several activities. Figure 1 shows main components and activities that we have identified for the support of data concern-aware service engineering. Typically, in order to expose data to DaaS, the data provider has to perform the *Wrapping Data* activity, which defines service operations for accessing and managing data resources, and the *Publishing Interfaces* activity, which publishes service operations and exposed data resources into service registries. These two activities are supported by a majority of tools for engineering DaaSs. The exposed data is then accessed by *Data Consumers* via the *Selecting Data* activity.

However, in order to provide also data concerns associated with data resources, other activities are needed to perform: the *Evaluating Data Concerns* activity is used to determine data concerns while the *Describing Data Concerns* activity will utilize evaluated concerns and configurations to determine data concerns to be published. Then, data concerns can be associated with service interfaces or data returned to data consumers in the *Selecting Data*. Furthermore, when the data is updated, in *Updating Data*, data concerns can be evaluated.

A. Wrapping, Selecting and Updating Data in DaaS

Wrapping Data, *Updating Data*, and *Selecting Data* are fundamental tasks in the development of DaaS and techniques to support them are well-developed. Generally, given a data source, the DaaS and data providers will expose certain data resources of the data source by allowing data resources to be accessed via service operations (e.g., based on WSDL or REST APIs). Using these operations, the data consumer will select certain data resources via service interfaces. Currently, main types of data to be exposed into DaaS are structured and unstructured data. The typical ways of accessing these types of data via DaaSs can be simplified in Figure 2:

- *structured data* is typically accessed by first mapping service operation parameters to data queries (based on the structure of the data) and then querying the content of the data. This popular way of accessing data typically relies on well-known query languages which are tightly coupled with the structure of the data. For example, structured

[5]In this paper we use the term *service provider* to indicate actors offering service features while *data provider* indicating actors offering data features. Therefore, the provider can also consider as "integrator", "creator", and "developer". However, the data provider does not mean that the provider has created the data but has the right to publish the data.

[6]http://infochimps.org/

[7]http://wso2.com/products/data-services-server/

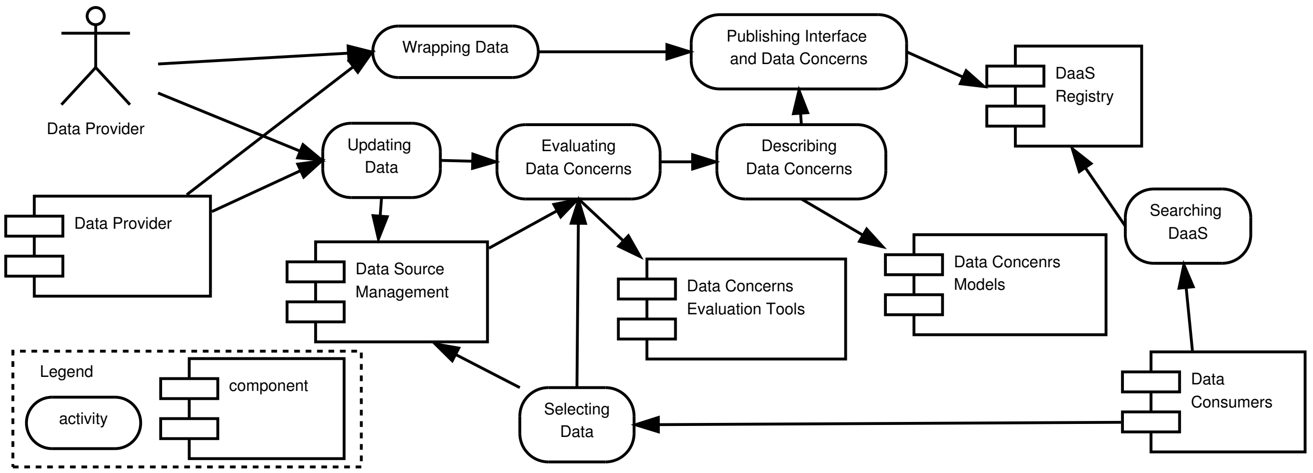
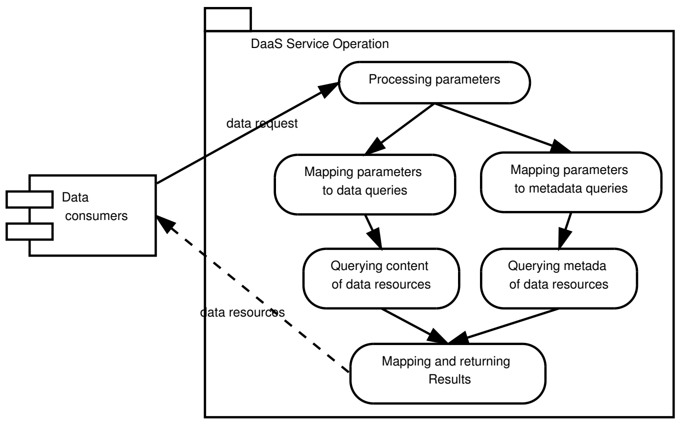


Fig. 1. Activities in the data concern-aware service engineering process. Arrow lines indicate control flows.

data in relational database is supported by mapping Web services operations to SQL and data query operations [4], [5] and by SOAP service wrapping to SQL database [6] while XML and RDF data can be accessed via Web services operations using XQuery/SPASQL.

- *unstructured data* is typically accessed by first mapping service operation parameters to metadata queries and then querying metadata to locate the data resources. The metadata can be, for example, semantic concepts, keywords and resource identifiers. Examples are REST APIs with parameter mapping to metadata for images [7] and for XML-based documents [8].



DaaS Service Operation
Processing parameters
data request
Mapping parameters to data queries
Mapping parameters to metadata queries
Data consumers
Querying content of data resources
Querying metada of data resources
data resources
Mapping and returning Results

Fig. 2. Main tasks implemented in a DaaS service operation for providing structured and unstructured data

In order to support the data concern-aware engineering process, first, we must be able to identify the data to be exposed and/or the data to be returned via *Selecting Data* and *Updating Data* activities. Second, this data identification, either data resource references and/or data resource values, must be passed to corresponding data concern evaluation tools which are invoked through instrumentation code inside DaaSs to determine possible data concerns. Third, evaluated data concerns can be published, monitored and composed by DaaS providers and data consumers.

B. Evaluating Data Concerns

Certain data concerns can be evaluated from the data, for example, quality of data (QoD) and data privacy metrics. The goal of the *Evaluating Data Concerns* activity aims at providing extra information about concerns for such data resources by evaluating data concerns based on the movement of data resources from DaaS to data consumers. Certainly, some data concerns can be determined before the data is exposed through DaaSs or before the data is accessed via DaaS operations, such as when the data is produced or is updated via the *Updating Data*. In these cases, several tools can determine data concerns and store them together with the data, for example, in the case of managing QoD in probabilistic databases [9]. In our approach we evaluate data concerns during the *Selecting Data* activity and consider pre-defined data concerns as inputs for tool-specific evaluations in our process.

In our process, three important aspects in evaluating data concerns are (i) the *scope* of the evaluation, (ii) the *mode* of the evaluation, and (iii) the *integration model* of evaluation tools with data answering.

1) Evaluation scopes: In our process, data concerns can be determined and evaluated based on three scopes:

- *data resource*: data concerns representing individual data resource(s). In most cases, the data consumer will access an individual data resource, such as a customer data record or a satellite image. Data concerns associated with individual data resources will help deciding whether a data resource should be used and under which conditions.
- *service operation*: data concerns representing all data provided by specific service operations. They help deciding which service operations should be used and when.
- *the service as a whole*: data concerns representing the data service as a whole will be useful for data concern-aware service discovery and selection.

In all scopes, data concerns can be determined by tool-specific implementations (e.g, by checking data quality and privacy of a customer record) and composition rules (e.g, using max and min operators for data quality metrics of a set of images).

2) Evaluation modes: Two modes for the evaluation of data concerns in our process are *off-line* and *on-the-fly*. In the off-line mode, data concerns can be determined before the data is accessed (e.g., data quality metrics of an image can be determined before any access to the image). On the on-the-fly, data concerns are evaluated for data movement requested through service operations (e.g., data accessibility is determined when an image is accessed). In some cases, data concerns are evaluated by using both off-line and on-the-fly. For example, static QoD metrics can be determined and unchanged for images. Then, static QoD metrics can be stored. However, QoD metrics for a set of images have to be determined on-the-fly, e.g., using QoD composition operators.

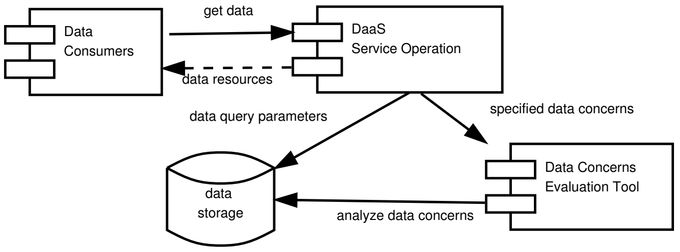


Fig. 3. Pull, pass-by-reference model for evaluating data concerns

3) Evaluation integration models: In our process, the integration model can be *pull* or *push*. In the pull model, DaaS operations will invoke *Data Concerns Evaluation Tool* by passing data resource references or data resource values. The pull model of passing data resource references for evaluating data concerns is shown Figure 3. In this model, the data is accessed as normally. In addition to that, data concerns can be separately evaluated (at the same time, before, or after the data access). It is easy to include new data concerns tools and to utilize pre-evaluated data concerns. This integration model can support generic data resources (e.g., relational database, XML database, file-based data). This can support on-the-fly evaluation and data concerns evaluation tools can be an external software-as-a-service as well. The pull, pass-by-value evaluation model is shown Figure 4. Within the DaaS operation, data is captured and specific tools are used to evaluate data concerns. This is suitable for specific data concerns (e.g., anonymity operation on privacy data) and requires a tightly coupling between DaaS operations and evaluation tools due to the passing of data values.

Figure 5 describes the push integration model. In this model, data is pushed to evaluation tools which determine concerns and pass the evaluated concerns to the DaaS service operation. This model is suitable for active data sources or subscribed data which is continuously pushed to the DaaS service operation.

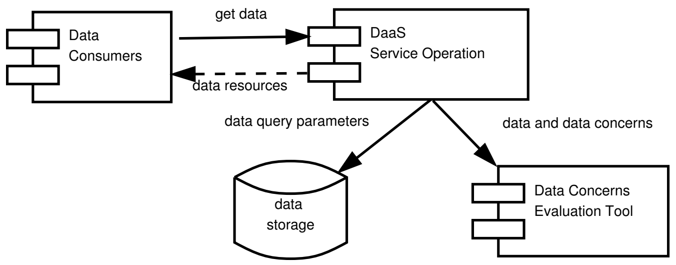


Fig. 4. Pull, pass-by-value model for evaluating data concerns

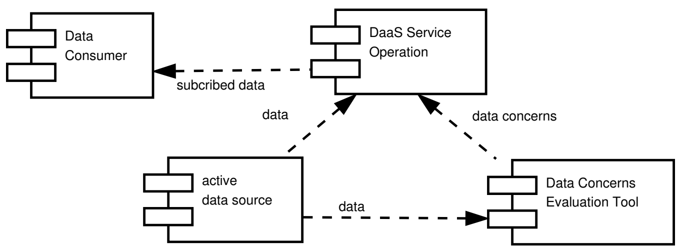


Fig. 5. Push model for evaluating data concerns of active data sources

C. Publishing Data Concerns Information

A specification consisting of several dimensions is needed in order to publish evaluated data concerns for DaaSs. In our work, the dimensions are (i) data concern category, (ii) evaluated and published time, (iii) evaluation scope, and (iv) concern provider information. As data concerns are evaluated with different evaluation scopes and modes, we support the following published models.

1) Off-line publishing of data concerns: the publishing of data concerns of a data resource is separated from the service operation which provides the access to the data resource. Therefore, typically data concerns of a data resource are evaluated and published before the data resource is actually requested by the data consumer. In this model, typically data concerns together with service information can be managed by service registries which provide links to service descriptions, data descriptions and data concerns. This kind of publishing is suitable for static data concerns.

2) On-the-fly publishing of data concerns by associating concerns with retrieved data resources: The resulting data resources (e.g., via queries) are annotated with data concerns evaluated by data concerns evaluation tools. This requires the data resources representation to be designed or to be capable of extension to include annotated data concerns. This kind of publishing is suitable for providing dynamic data concerns.

3) On-the-fly publishing of data concerns through queries: this method supports the use of different service operation parameters to query data concerns of data resources. It works in a similar manner to the request of a data resource. However, instead of returning a data resource, the service operation will return the data concern evaluated for the requested data resource on-the-fly. This kind of publishing is suitable for validating data concerns before accessing data resources (e.g., in case of large volume of data).

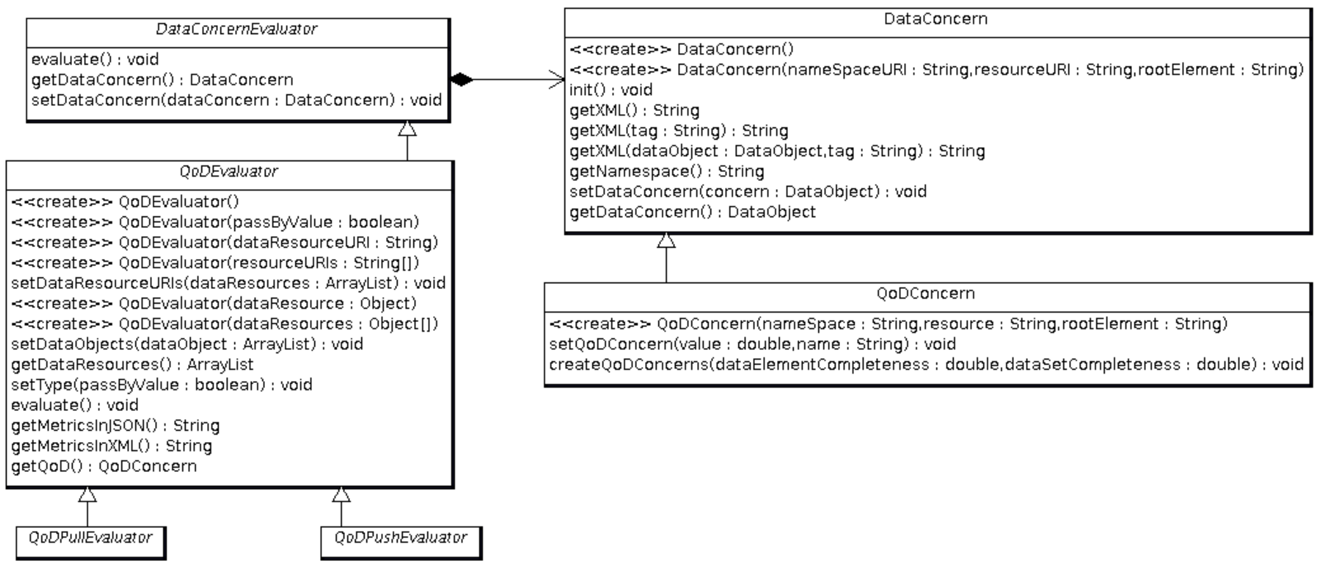

DataConcernEvaluator
evaluate() : void
getDataConcern() : DataConcern
setDataConcern(dataConcern : DataConcern) : void
DataConcern
<<create>> DataConcern()
<<create>> DataConcern(nameSpaceURI : String,resourceURI : String,rootElement : String)
init() : void
getXML() : String
getXML(tag : String) : String
getXML(dataObject : DataObject,tag : String) : String
getNamespace() : String
setDataConcern(concern : DataObject) : void
getDataConcern() : DataObject
QoDEvaluator
<<create>> QoDEvaluator()
<<create>> QoDEvaluator(passByValue : boolean)
<<create>> QoDEvaluator(dataResourceURI : String)
<<create>> QoDEvaluator(resourceURIs : String[])
setDataResourceURIs(dataResources : ArrayList) : void
<<create>> QoDEvaluator(dataResource : Object)
<<create>> QoDEvaluator(dataResources : Object[])
setDataObjects(dataObject : ArrayList) : void
getDataResources() : ArrayList
setType(passByValue : boolean) : void
evaluate() : void
getMetricsInJSON() : String
getMetricsInXML() : String
getQoD() : QoDConcern
QoDConcern
<<create>> QoDConcern(nameSpace : String,resource : String,rootElement : String)
setQoDConcern(value : double,name : String) : void
createQoDConcerns(dataElementCompleteness : double,dataSetCompleteness : double) : void
QoDPullEvaluator
QoDPushEvaluator

Fig. 6. Main classes for evaluating and publishing QoD metrics

IV. A Framework for Evaluating and Publishing QoD of DaaSs

Based on the process presented in Section III, we have developed a framework for evaluating and publishing QoD associated with data offered by DaaS.

A. *Pull QoD Evaluation Models for DaaS*

To support pull models with pass-by-references and pass-by-value for QoD evaluation, we use the following ways to describe references and values of data resources:

- reference of data resources: a reference of data resource is an URI. In our work, data resources URI can be known only within the DaaS offering data resources or known by any data consumers.
- value of data resources: a value of data resources passed to an evaluation tool can be any object.

Any reference and value passed to a specific evaluation tool must be understood by the tool and this constraint is not a concern of our framework. The ways to describe references and values of data resources can be of course dependent on different frameworks followed our data concern-aware service engineering process in Section III.

In our framework, quality of data tools are developed by third parties and are plugged into our framework. Each tool will be configured suitable for data concerns, and evaluation scopes, models and integration models. In the literature, several QoD metrics and frameworks for evaluating these metrics exist [10]. Figure 6 describes generic classes for QoD evaluators wrapping generic or specific QoD evaluation tools. `DataConcernEvaluator` defines a generic abstract class for any data concern evaluators. `QoDEvaluator` provides an abstract class for QoD evaluators. Whether a `QoDEvaluator` is a pass-by-value or pass-by-reference can be controlled by the `setType` method. Appropriate references and values passing must be used by using corresponding methods of `setDataResources` and constructors. The `evaluate` method is tool-specific implementation and it can also wrap external tools or software-as-a-service performing the real evaluation. The result of a QoD evaluation is described using the `QoDConcern` class which allows extensible QoD metrics to be modeled. `QoDConcern` is a specific type of `DataConcern` which is a generic class describing data concerns. We implement a generic way of describing QoD metrics by extending the data concern model in [2].

B. *QoD Publishing*

All publishing models in Section III-C are supported. For off-line publishing of data concerns, we devise a new model of publishing that cover three levels of evaluation scopes. Figure 7 describes main classes of our common data concern publishing specification. The three scopes of data concerns are described by `Service`, `ServiceOperation`, and `DataResource`. For each of these scopes, publishing data concerns are stored in `Entry` which uses a `EntryContent` to represent concrete data concerns inline or uses an external link to indicate data concerns in an external source. Our publishing specification allows any types of data concerns modeled in different representations to be included into `EntryContent`. We have developed a tool that can invoke QoD evaluators and publish evaluated QoD concerns for DaaS based on this specification.

C. *QoD Monitoring and Composition*

QoD concerns monitoring and composition are required by DaaS providers to publish QoD metrics associated with data resources as well as by DaaS consumers to evaluate aggregated data resources. To support the monitoring of QoD

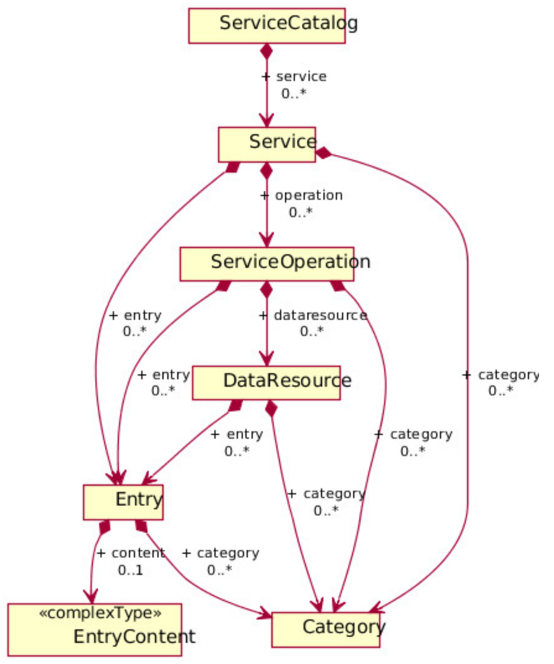


Fig. 7. Overview of the data concern publishing specification

metrics, we allow DaaS providers and consumers to invoke monitoring rules by passing QoD metrics of data resources to a rule engine. Similarly, to support the composition of QoD metrics, QoD metrics can be fed from evaluators within service operations or from the use of queries for obtaining data concerns on-the-fly. Rules are then can be associated with workflows for composing data. We use Drools[8] to implement QoD monitoring and composition rules and workflows.

V. Experiments

We have implemented our framework using Java and JAX-RS/Jersey[9]. We utilized some published datasets at the UN-DataAPI [8]. Currently, each dataset can be downloaded as an XML file via its identifier. However, neither any dataset nor the UNDataAPI includes QoD metrics.

First of all, let assume a data provider wants to publish QoD associated with these datasets and the data provider implements her own DaaS (thus in this case the DaaS provider is the same to the data provider). By using existing quality of data definitions and techniques [10], we implemented a specific QoD evaluator named `QoDUNDataEvaluator` based on `QoDEvaluator`. `QoDUNDataEvaluator` is a *pull, pass-by-value* evaluator (see Figure 4 and Section IV-A) and evaluates `datasetcompleteness` and `dataelementcompleteness` metrics, which describe the completeness of the list of countries in the period of 1990–2009 and the completeness of the data elements of that list in that period, respectively. We implemented a RESTful DaaS acting as a proxy to the UNDataAPI service so that these datasets can be accessed via our DaaS service while our service performs the evaluation of data concerns.

Listing 1 shows a (simplified) implementation of a RESTful service operation that provides (i) only the requested data resource, (ii) only QoD metrics, and (iii) the requested data resource annotated with QoD metrics. Lines 5-10 show a typical DaaS implementation: based on the id of the requested data resource, the DaaS returns the requested data. In this example, we obtained the data and processed it using a `DataObject` of the Service Data Objects (SDO) concept [11]. Lines 12-17 show how we evaluated QoD metrics of the requested data resource and returned only the evaluated QoD metrics using the `QoDUNDataEvaluator`. Here we used a convention that when the data consumer specifies a query parameter `QoD` without a value, the only the QoD metrics are requested. Lines 19-27 show the case of providing the requested data resource and QoD metrics when the data consumer specified `QoD=annotation` in the REST operation. We evaluated QoD metrics and by using SDO techniques, we annotated the QoD metrics with the requested data resources. The example in Listing 1 demonstrated the usefulness of our methodology and our implementation framework to provide different ways to access not only the requested data resources but also their QoD metrics on-the-fly. In particular, the technique shown in Lines 19-27 can be applied to the evaluation of data concerns for active DaaSs which push data resources to data consumers.

Second, let us assume that the data provider just utilizes an existing DaaS (e.g., similar to Infochimps) to publish her data but she needs to include also data concerns. In this case, the DaaS service provider can utilize our common data concern publishing specification and tool and allows the data provider to specify her data concerns (as the DaaS service provider is not able to evaluate the data concerns in this assumption). Listing 2 shows an excerpt of the information evaluated and published for the dataset `Adult literacy rate` using our specification and tool. This publishing information can be managed by service registries or provided via Atom feeds by the DaaS and data providers.

Third, let us assume that a data consumer, which is also a data integrator, wants to compose two data sources `Adult literacy rate` and `Population annual growth rate` in order to examine the correlation between literacy and population growth rates. This consumer is particularly concerned about the QoD of data resources to be composed. Using our previous techniques, the consumer can easily obtain the QoD metrics associated with these data resources provided by DaaSs. By using QoD metrics, the data consumer can define different rules to monitor the QoD of data resources as well as to derive new QoD metrics for composite data. Listing 3 shows two simple rules using Drools. The rule `LowQuality` indicates that if either `datasetcompleteness` or `dataelementcompleteness` of any data resource is less than 0.5, a pre-defined threshold, than the data to be composed is of low quality. The rule `MinComposition` defines a simple way to determine QoD metrics for a composite data based on the minimum QoD metrics of data resources to be

[8]http://jboss.org/drools

[9]Due to space limit, we provide supporting materials under http://www.infosys.tuwien.ac.at/prototyp/SOD1/dataconcerns/dcevaluation.html

```
@GET
@Produces("application/xml")
public String getXml(@PathParam("id") String id, @QueryParam("QoD") String QoD) throws Exception {
    //...
    DataObject dataobject = getDataResourceByID(id);
    //...
    //return the requested data resource only
    if (QoD == null) {
        return toXML(dataobject, "http://www.undata-api.org", "results");
    }
    //evaluate and return only the quality of data of the requested data resource
    if (QoD.isEmpty()) {
        QoDUNDataEvaluator qodEval = new QoDUNDataEvaluator(dataobject);
        qodEval.setType(true);
        qodEval.evaluate();
        return qodEval.getMetricsInXML();
    }
    //evaluate and return the requested data resource and its quality of data
    if (QoD.equals("annotation")) {
        QoDUNDataEvaluator qodEval = new QoDUNDataEvaluator(dataobject);
        qodEval.setType(true);
        qodEval.evaluate();
        DataObject resultObject = dataobject;
        //...
        resultObject.getSequence().add("qod",qodEval.getDataConcern().getDataConcern());
        return toXML(resultObject, "http://www.infosys.tuwien.ac.at/SOD1/undata-api", "results");
    }
    //...
}
```

Listing 1. Code excerpt for evaluating and providing a requested data resource with/without its quality of data

```
<dcp:dataresource xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:dcp="http://www.infosys.
    tuwien.ac.at/prototype/SOD1/dcp" xsi:type="dcp:DataResource">
  <dcp:category label="Demographic and socioeconomic statistics"/>
  <dcp:id>http://undata-api.appspot.com/data/query/Adult literacy rate (percent)</dcp:id>
  <dcp:title>UN Data on Adult literacy rate</dcp:title>
  <dcp:entry>
    <dcp:category term="DaaSConcerns"/>
    <dcp:content type="application/xml">
<tns:qod xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:tns="http://www.infosys.tuwien.ac.at/
    SOD1/daasconcerns/custom" xsi:type="tns:QoD">
  <tns:dataelementcompleteness>0.5874439461883408</tns:dataelementcompleteness>
  <tns:datasetcompleteness>0.04349775784753363</tns:datasetcompleteness>
</tns:qod>
</dcp:content>
    <dcp:published>2010-06-23T22:23:30.557Z</dcp:published>
  </dcp:entry>
</dcp:dataresource>
```

Listing 2. Example of QoD metrics with data resources

composed. Although these rules are simple examples, with a rich set of QoD metrics for data resources and rules, data consumers can build strong quality-aware solutions when using DaaSs in the Internet and cloud environments.

VI. Related work

Quality of data (or quality of information) becomes an important issue on the Internet and cloud environments [3]. However, little effort have been spent on supporting the evaluation, management and publishing of data concerns associated with data offered by DaaSs. Techniques for engineering data services have discussed in several places [4] and quality of data in general and in database is a popular research with many well-researched frameworks and tools [10]. However, existing works do not address the integration of data concerns evaluation with DaaS implementation and DaaS information publishing. Our work does not deal with particular quality of data assessment techniques but provides a generic data concern-aware service engineering process and a generic framework integrating and utilizing existing QoD techniques through wrappers.

Several models and frameworks have been developed for publishing information about Web services but they neglect the publishing of data concerns. Unlike QoS publishing [12] which is typically done at the service as a whole level, data concerns are strongly associated with specific data resources. As a result, data concerns cannot just be described as the level of service as a whole. Our publishing model is much richer

```
rule "LowQuality"
when
  not (forall ($qodConcern: QoDConcern((datasetcompleteness > 0.5)&&(dataelementcompleteness > 0.5))))
then
  System.out.println("The quality of data is too low");
end
rule "MinComposition"
when
 forall ($qodConcern: QoDConcern((datasetcompleteness > 0.5)&&(dataelementcompleteness > 0.5)))
 $mindatasetcompleteness : Double()
   from accumulate( QoDConcern($datasetcompleteness : datasetcompleteness , $dataelementcompleteness :
       dataelementcompleteness )
   min($datasetcompleteness) )
 $mindataelementcompleteness : Double()
   from accumulate( QoDConcern($dataelementcompleteness : dataelementcompleteness )
   min($dataelementcompleteness) )
then
 System.out.println("Minimum datasetcompleteness: " +$mindatasetcompleteness);
 System.out.println("Minimum dataelementcompleteness:" +$mindataelementcompleteness);
end
```

Listing 3. Simple rules for monitoring and composing QoD metrics

by covering different scopes in off-line and on-the-fly modes.

In [13], an approach for managing quality of data associated with information products is presented. Essentially, that work presents the movement of information product through different places and which are possible metrics associated with information products. In our view, it is related to how quality of data can be internally evaluated inside DaaS. If such information product already has quality, then using our model, both information product (data in our DaaS concept) and its quality can be accessed via DaaS. [14] shows how to manage quality of data using Web services. Our off-line data concern publishing information can be managed by Web services. However, we also provide on-the-fly data concerns publishing. Furthermore, data resources and their data concerns can be accessed in the same DaaS.

VII. Conclusion and Future Work

Data concerns associated with data provided by DaaS should be accessible in order to support the selection and composition of data. In this paper we present a novel data concern-aware service engineering process for DaaS and a generic framework, as an implementation of our process, to support the evaluation and publishing of quality of data metrics associated with data exposed through DaaS. Our process and framework is able to support different evaluation scopes, modes and integration models. Our integration of the evaluation of data concerns within DaaS with the publishing of data concerns for data movements covers both off-line and on-the-fly data concern publishing.

Our process is generic enough to be applied to different data concerns, although in our particular framework we presented only QoD metrics. In principle, using our framework, a data consumer can also perform the evaluation of data concerns for data provided by DaaS. In our future work, we will extend our experiments to cover different evaluation scenarios and richer data resources and to evaluate the performance of data concern evaluation. Furthermore, we plan to extend the monitoring and evaluation of data concerns for data composition and work on the dependency between the context of data customers and data concern evaluation.

References

[1] A. Dan, R. Johnson, and A. Arsanjani, "Information as a service: Modeling and realization," *Systems Development in SOA Environments, 2007. SDSOA '07: ICSE Workshops 2007. International Workshop on*, pp. 2–2, May 2007.

[2] 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.

[3] E. Bertino, A. Maurino, and M. Scannapieco, "Guest editors' introduction: Data quality in the internet era," *IEEE Internet Computing*, vol. 14, pp. 11–13, 2010.

[4] F. Zhu, M. Turner, I. Kotsiopoulos, K. Bennett, M. Russell, D. Budgen, P. Brereton, J. Keane, P. Layzell, M. Rigby, and J. Xu, "Dynamic data integration using web services," in *ICWS '04: Proceedings of the IEEE International Conference on Web Services*. Washington, DC, USA: IEEE Computer Society, 2004, p. 262.

[5] Z. Wei, J. Dejun, G. Pierre, C.-H. Chi, and M. van Steen, "Service-oriented data denormalization for scalable web applications," in *WWW '08: Proceeding of the 17th international conference on World Wide Web*. New York, NY, USA: ACM, 2008, pp. 267–276.

[6] T. Malik, A. S. Szalay, T. Budavari, and A. Thakar, "Skyquery: A webservice approach to federate databases," *CoRR*, vol. cs.DB/0211023, 2002.

[7] "Flickr Service APIs, http://www.flickr.com/services/api," last access: July 15, 2010.

[8] "The UNDATA API project, http://www.undata-api.org/," last access: July 15, 2010.

[9] N. Dalvi, C. Ré, and D. Suciu, "Probabilistic databases: diamonds in the dirt," *Commun. ACM*, vol. 52, no. 7, pp. 86–94, 2009.

[10] C. Batini, C. Cappiello, C. Francalanci, and A. Maurino, "Methodologies for data quality assessment and improvement," *ACM Comput. Surv.*, vol. 41, no. 3, 2009.

[11] "Service Data Objects Specifications," http://osoa.org/display/Main/Service+Data+Objects+Specifications, last access: 3 August 2010.

[12] S. Ran, "A model for web services discovery with qos," *SIGecom Exch.*, vol. 4, no. 1, pp. 1–10, 2003.

[13] G. Shankaranarayan, M. Ziad, and R. Y. Wang, "Managing data quality in dynamic decision environments: An information product approach," *J. Database Manag.*, vol. 14, no. 4, pp. 14–32, 2003.

[14] Y. Cai and G. Shankaranarayanan, "Managing data quality in inter-organisational data networks," *IJIQ*, vol. 1, no. 3, pp. 254–271, 2007.