DEMODS: A Description Model for Data-as-a-Service

Quang Hieu Vu*, Tran-Vu Pham†, Hong-Linh Truong‡, Schahram Dustdar‡, Rasool Asal*
*ETISALAT BT Innovation Center, Khalifa University, UAE, quang.vu@kustar.ac.ae; rasool.asal@bt.com
†Faculty of Computer Science and Engineering, HCMC University of Technology, Vietnam, t.v.pham@cse.hcmut.edu.vn
‡Distributed Systems Group, Vienna University of Technology, Austria, {truong,dustdar}@infosys.tuwien.ac.at

Abstract—**Cloud computing based Data-as-a-Service (DaaS) has become popular. Several data assets have been released in DaaSes across different cloud platforms. Nevertheless, there are no well-defined ways to describe DaaSes and their associated data assets. On the one hand, existing DaaS providers simply use HTML documents to describe their service. This simple way of service description requires user to manually perform service lookup by reading the HTML documents to understand DaaSes as well as their provided data assets. On the other hand, existing service description techniques are not suitable for describing DaaSes because they consider only service information. The lack of well-structured/linked model to describe DaaSes hinders the automatic service lookup for DaaSes and the integration of DaaSes into data composition and analytic tools. In this paper, we propose DEMODS, a DEscription MOdel for DaaS, which introduces a general linked model to cover all basic information of a DaaS. Besides the basic DaaS description model, we also introduce an extended model that integrates existing work in describing quality of data, data and service contract, data dependency, and Quality of Service (QoS). We present a mechanism to incorporate DEMODS into both new and existing DaaSes. Finally, a prototype of DEMODS has been developed to evaluate the effectiveness of the proposed model.**

Keywords - *Cloud computing, Data-as-a-Service (DaaS), data marketplace, service and data description, discovery.*

I. Introduction

Data-as-a-Service (DaaS) is a type of cloud computing services that provide data on demand. A DaaS typically exposes its provided data to consumers through APIs, either for the consumer to download or query data from different data assets. By using DaaSes, consumers do not need to fetch and store giant data assets and search for the required information in the data asset. Instead, they simply find a suitable DaaS that provides the data asset having the desired information and call the corresponding APIs to retrieve the data. With recent developments in cloud computing, it has become easier to build DaaSes that provide bigger data assets at lower costs on the clouds. Since the last couple of years we have witnessed the rapid growth in the number of DaaSes. Typical DaaS providers support from generic data assets, such as Amazon [1], Microsoft Azure Data Marketplace [2] and Infochimps [3], to specific data assets, like Gnip [4].

While there have been existed in the market a number of DaaSes, there is no well-defined structured model in the description of their services. Each DaaS has a unique way to describe its provided service as well as supplied data assets. For example, in the DaaS provided by Amazon [1], data are stored in snapshots of EBS (Elastic Block Store), where only a list of data assets available in the service, their short description, and links to data asset providers are given. In this way, if a user wants to know more about the data assets, he/she needs to visit the link of data asset providers. On the other hand, Microsoft Azure Data Marketplace provides more information in its DaaS [2], e.g., by describing in detail fields in provided data assets in addition to their general description. Despite the difference, both Amazon and Microsoft require their users to manually read HTML documents to understand provided services and data assets. This disadvantage strongly hinders the discovery of services as well as data assets based on complex data concerns.

In this paper, we aim to address the lack of well-structured models in DaaS description by proposing DEMODS – a DEscription MOdel for DaaS – as a conceptual information model for DaaSes. Our description model includes basic information of a DaaS to describe the DaaS itself, its general APIs, available data assets in the DaaS as well as APIs for querying data from the data assets, and pricing models associated with both data assets and the DaaS. Besides the basic model, we also introduce an extended model that integrates existing work, e.g., in data contract, data dependency, and quality of service. Finally, we show how to employ DEMODS in existing DaaS service engineering techniques in order to provide rich service information for DaaS service discovery and lookup. In summary, our paper makes the following major contributions.

- We propose both a basic version and its extended one of DEMODS, a description model for DaaS.
- We introduce a mechanism to incorporate DEMODS in both existing and new DaaSes.
- We develop a prototype as a proof-of-concept for the proposed model and show how useful the model is with concrete examples.

The rest of the paper is organized as follows. In Section II, we present our motivation and related work. In Section III, we discuss all aspects of a DaaS that needs to be addressed in its description. In Section IV, we present DEMODS, our DaaS description model. In Section V, we discuss the integration between DEMODS and DaaS. We describe an implementation of a prototype for the proposed model in Section VI. Finally, we conclude the paper and discuss our future work in Section VII.

II. Motivation and Related Work

A. Motivation and Approach

Our work is motivated by the lack of a rich information model for DaaS in data marketplaces. On the one hand, even though DaaS plays a central role to deliver data from data providers to data consumers in data marketplaces, existing DaaS providers have their own way to describe DaaSes, mostly using HTML documents. On the other hand, existing service description techniques are not adequate in supporting description for DaaS, as data assets in data marketplaces are associated with different properties and there is a clear separation of information about provided services and supplied data assets. This lack of information model leads to three basic drawbacks as follows.

- Service and data discovery cannot be done automatically. When a data consumer wants to find specific data assets, specific data in data assets, or specific types of DaaSes, he/she has to visit DaaS providers one by one to look for their provided service and data asset description. This manual way is time-consuming and mostly suitable for human beings. Furthermore, given the number of DaaS providers as well as data assets could be very large, this way of discovery is tedious.
- On-demand data integration, service integration, and query optimization cannot be supported. While data integration is desirable when data consumers want to retrieve data from different DaaSes, service integration is needed when users want to use services from different DaaS providers. On the other hand, query optimization is important when different DaaSes, for example, set different prices and data agreement and have different quality for the same type of data or data service. Manual discovery of DaaSes and data assets does not foster on-the-fly data/service integration and query optimization.
- Service/data information and DaaS engineering cannot be tied. In particular, service description and service implementation are currently independent on each other, and hence there could be inconsistency between what is described in HTML documents and what is actually implemented in DaaSes.

Our approach to address these drawbacks is to develop a unified, yet scalable, information model for DaaS. Partially, our DEscription MOdel for DaaS (DEMODS) is based on linked data principles in which different types of data could be linked to. The role of DEMODS and how useful it is in an ecosystem of a DaaS as well as in an ecosystem of a data marketplace are depicted in Figure 1. As illustrated, DEMODS could be used at both the service and data discovery level outside DaaSes to support automatic service and data discovery and the service implementation level inside DaaSes to support data integration, query optimization, and consistency among service description, payment, and authorization processes. In particular, by leveraging DEMODS, automatic service and data discovery can help data consumers to find services as well as data assets according to several criteria easily. On the other hand, with the sharing knowledge of DEMODS, it is easy to maintain consistency among different components in such ecosystems. Furthermore, semantic data integration could be done to collect data from different data assets and query optimization could be performed across multiple DaaSes.

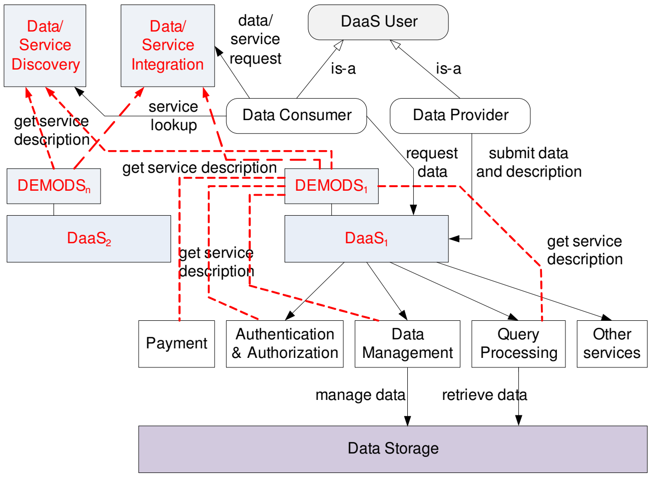


Fig. 1. DaaSes, users, and the use of DEMODS in a data marketplace

B. Related Work

In Service Oriented Computing, particularly Web Services technologies, there exist different types of service descriptions for various purposes. The well-known Universal Description Discovery and Integration (UDDI) was designed for registering and locating Web-based business services in business directories by describing basic information about Web Services, such as provider, contact, address, and functionality. Technically, a Web service is commonly described by Web Service Description Language (WSDL). WSDL and its extensions, such as SAWSDL [5], have the mechanism for describing service interfaces and bindings that are necessary for invoking a Web service. To enable automatic discovery and invocation of Web services, there have been a number of ontology-based service description models introduced, such as DAML-S [6] and OWL-S [7]. For managing Web service evolution, SEMF [8] was introduced. The Unified Service Description Language (USDL) was ambitiously introduced for both technical and business services [9].

Despite the abundance of web service description models, none of them is good enough for describing DaaSes due to one main reason: they can only model information about services, their functional capabilities, interaction mechanisms, etc., but are not able to describe multiple types of data assets delivered by DaaSes. Descriptions about data assets in DaaSes not just include common types of metadata, such as data size, data provider, and data schemas in contemporary data catalogs [10], [11], but also consist of, e.g., APIs for accessing data assets, data contract, and pricing models. SEMF, for example, links different types of service information, including also service licensing. However, SEMF considers description only at the service level, thus it cannot be used for modeling data assets within services. The USDL supports different service information, covering technical interface, service capabilities, pricing, legal, etc, but at the service level only. For this reason, we develop DEMODS particularly for describing DaaSes. DEMODS includes information about services at service level and also information about data assets delivered by DaaSes such as data category, data agreement and licensing, and quality of data. DEMODS can be used with other existing service description models such as WSDL, service agreement model [12], service licensing model [13], etc. At the data asset level, DEMODS is not designed to replace metadata models used for describing data assets delivered by DaaSes. Instead, DEMODS provides a mechanism for integrating with those models. For example, DEMODS can be linked with the Open Data Protocol (for publishing information about data structure) [14], or Dublin Core (general purpose metadata) [15].

III. DaaS Service Information Characteristics

Before introducing a general information model for DaaS description, it is necessary to analyze basic characteristics of DaaSes. In this section, we will focus on a general classification of DaaSes, basic levels of DaaS description, types of information used in DaaSes, and DaaS description in relation to DaaS engineering.

A. Types of DaaS

In general, we observe three types of DaaSes – depending on the number and relationships of data assets in their services: generic, specialized, and hybrid

- Generic DaaS category: DaaSes in this category provide several independent data assets in their services. They usually support operations for accessing individual data assets as a whole as well as internal parts of data assets. As a result, service description is mainly associated with data assets and APIs for data assets. Examples include Microsoft Azure Data Marketplace [2], Infochimps [3], Amazon Public Datasets [1], and Socrata [16].
- Specialized DaaS category: DaaSes in this category only provide data based on a single data asset or a limited number of related data assets in their services. They often support operations to work on data assets collectively, instead of individual data assets, and hence service information focuses on describing service as a whole and service APIs. Examples include Gnip [4] and networks of bio-diversity data assets.
- Hybrid DaaS category: In the hybrid model, a generic DaaS provides data assets from many different specialized DaaSes on their behalf, together with its own data assets. Access to data assets within the hybrid DaaS is done in two steps: (i) identify location of the data assets (whether the data assets are of the generic DaaS or specialized DaaSes) (ii) retrieve the data assets using the generic DaaS APIs or the APIs of the specialized DaaS which provides the data assets. Therefore, service information needs to include the information about the generic service itself, its APIs, and provided data assets together with information about specialized DaaSes within it.

B. Structure and description levels of DaaS

DaaSes in the three different categories of DaaSes described in the previous section can be generalized in a unified structure as follows. For each DaaS, there exists a generic set of APIs for its operations, such as for searching, downloading, uploading, and updating data assets. A DaaS operates on a number of data assets and a set of specialized DaaSes. For operations on specific data assets, e.g. querying or retrieving part of the data asset, each data asset has its own APIs. Similarly, for operations on specialized DaaSes, each DaaS is also associated with a specific set of APIs. As specialized DaaSes operate directly on some specific data assets, specialized DaaS APIs can be used to manipulate internal structure of data assets it provides. Note that data assets might not be static but dynamically updated by data providers.

Based on the analysis of DaaSes described in the previous subsection, we observe that there exist two main levels of DaaS description.

- Service level: provides general description about DaaS such as provider, name, identification, etc. From operational point of view, operations at Daas level, such as discovery, data asset/service registration, etc., are supported by DaaS service level APIs. These APIs concentrate on retrieving data from multiple data assets and are often found in specialized DaaSes.
- Data asset level: includes information specific to particular data assets. Operations on data assets are supported by data asset level APIs or specialized DaaS APIs. These APIs focus on retrieving data from individual data assets.

C. DaaS information types

In addition to identifying levels of description as discussed in the previous section, it is important to identify common types of information in DaaSes. Without having common information types, each DaaS may define its own types of information, and hence it becomes impossible to support data integration and query optimization across DaaSes. In particular, with respect to data integration, it is necessary to have a common agreement in the definition of data domains and data categories (e.g., if a data asset is classified in the category of bio-medical, what type of data is expected in the data asset?). On the other hand, with respect to query optimization, common agreed pricing models defining how users are charged for using data and service are expected (e.g., if a DaaS is charged as pay-per-use, what does it mean?). Besides these common information types to support data integration and query optimization, it is worth to note that there could be other types of common information that are useful for advanced service lookup such as compliance laws, privacy information, and QoS information types.

D. DaaS description information in DaaS service engineering, management and discovery

DaaS description has different roles in DaaS service engineering, management, and discovery. While the ultimate goal of having an integrated DaaS description model is to

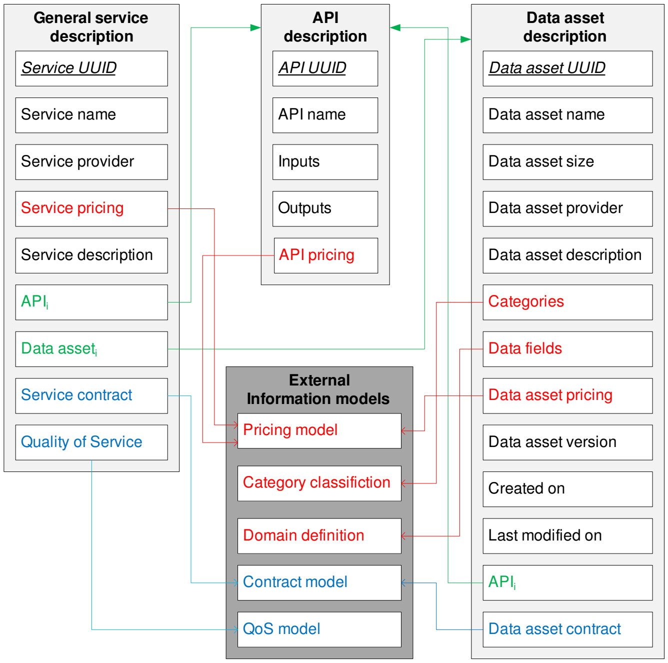


Fig. 2. An overview of DEMODS

have enough information to support many tasks (e.g., data discovery and composition), the model can have a strong impact also on service engineering and management within DaaSes. Specifically, from the service engineering perspective, at the moment, DaaSes could be implemented as either SOAP or REST with only information about service and service API available. Other information is currently gathered in different and specific ways, where there is no good support. As a result, by having an integrated model, we aim at improving contemporary service engineering processes for DaaSes. A trivial example is to leverage DaaS description to maintain consistency across different components from service management, account management, to cost management.

Furthermore, from service management and discovery perspective, big DaaSes or small/network of DaaSes have different ways to collect and manage different types of DaaS information that should be studied. In parallel with an integrated DaaS description information model, we aim at analyzing and proposing techniques for collecting and managing DaaS description information.

IV. Conceptualizing DaaS Service Description

In this section, we will introduce DEMODS, our proposed model for capturing description of DaaS. Figure 2 presents our DEMODS that is developed based on the concept of linked data in which different types of description, represented in different models, can be linked to characterize the description of DaaS at the service and the data asset levels. The *General Service Description* represents DaaS information at the service level and the *Data Asset Description* presents the description of data assets within DaaSes. Both levels of description have links to several external models classified into *API Description* and *External Information Models* for services and data assets.

A. General Service Description

An important part of the service description is to describe the service itself without considering published data assets. In general, basic information of a service includes service universally unique identifier (UUID), service name, service provider, service pricing models (i.e., the cost for using the service), and service description. Additionally, service information may include general APIs that are used to access data assets at a high level. For example, a DaaS may provide a general API called *get(data-asset, data-field)*, which retrieves data of the specified data-field in the input data asset. It is interesting to note that different from conventional API description, the description for APIs in DaaS may have an associated cost (i.e., the cost of using or calling APIs). This is a special feature of DaaS that allows users to have several options to pay for the service (e.g., by monthly plan or pay-as-you-go through every API call). We will elaborate different payment schemes of DaaS later in Section VI-A. The *General Service Description* component in Figure 2 illustrates basic information at the service description level.

B. Data Asset Description

While basic information of DaaS is described at the service description level, which can be used for service discovery, integration, and utilization, all important information of published data assets are defined at the data asset description level. As shown in the *Data asset description* component in Figure 2, each data asset provided by a DaaS is associated with general information about its UUID, its name, its size, its provider, its version, created date, last modified date, and its description. Additionally, to support better service lookup, it is desirable to have a description of categories the data asset belongs to as well as fields in the data asset. Finally, similar to the description at the service level, data asset description also consists of APIs and the specification of data asset usage cost (e.g., the cost of downloading the whole data asset). It is important to note that the APIs in the data asset description level are low-level APIs that access data within the data assets. On the other hand, the APIs in the service description level are high-level APIs that can access different data assets as well as multiple data assets at once.

C. External Information Models

Linking to outside external information models (or external ontologies) is important to DEMODS to support semantic service lookup. As shown in Figure 2 and Table I, there are a number of description items that should not be defined internally. Instead, these items should be open for definitions from outside, or in other words they could be linked to outside models (or ontologies). These items include categories of data assets that link to *Category classification*, fields in data assets that link to *Domain definition*, and costs of the service, data assets, as well as APIs that link to *Pricing model*. As pointed in [24], [25], ontologies are typically used as a way to support data integration and semantic services. Furthermore, external information models give DEMODS flexibility to

Model	Description	Examples
Pricing	describe how to charge consumers for using DaaSes and data assets.	General QoS ontologies and service agreements specifications usually allow to describe simple pricing information [17], [18], [12]
Category classification	define categories to which DaaSes and their data assets belong. This can be done by using category specifications or tagging.	SNaP Ontologies [19], Upper Tag Ontology [20], NAO [21]
Domain definition	define and describe the structure of data provided by DaaSes by indicating whether there exist specific specifications for describing structure data or for annotating information about data structure into data assets	OData [14] (for publishing data structure information) and OpenAnnotation [22] (for annotating data with metadata about data structure)
Contract	specify service/data contracts associated with DaaSes and with data assets	Web services Level Agreement (WSLA) [12], Service Licensing [13], the Abstract Data Contract Model [23]
QoS	specify the quality capabilities of DaaSes	QoS ontology [17], [18]

TABLE I
EXTERNAL INFORMATION MODELS LINKED INTO DEMODS

support different types of data assets from different domains and different requirements for services and data assets. For example, a data marketplace for data assets in the news and media domain may select the SNaP ontologies as a way to categorize its data assets, but such ontologies are not suitable for geodata. In DEMODS, by allowing external information models to be used, different data marketplaces may simply select suitable models to support by creating links to these models. Moreover, even if a service has special requirements in data usage, data concerns as in [26], or Quality of Service (QoS), these requirements could be easily specified as models outside and then are linked to DEMODS through external information models.

In DEMODS, possible external models (or ontologies) and their description are presented in Table I. Among external models, *Pricing, Category classification* and *Contract* are needed at both service and data asset levels. On the other hand, *QoS* is solely applied at the service level while *Domain definition* is only applied at the data asset level. In our view, *Category classification* or *Contract* can be treated separately for the service level and for the data asset level, because in terms of classification or contract, service and data have different taxonomies or contractual conditions. However, *Pricing* should not be treated separately since common pricing plans can be applied for both service and data. However, existing pricing description models either cover only service pricing (e.g., pay-per-transactions) or data pricing (e.g., pay-per-the quantity/quality of data). Therefore, we need to combine these models for describing pricing plans for DaaSes. In general, while we propose DEMODS as a basic description for DaaS, it is possible to extent it to support other aspects of DaaS by creating an extra annotation for the description and link it to an outside topology (e.g., privacy ontology).

V. DEMODS AND DAAS INTEGRATION

To some extent, DaaS can be considered as a special type of web service in which data is provided on demand. Therefore, DaaSes can be implemented as WSDL- or REST-based Web services, as WSDL and REST are the most popular technologies for services. To allow DEMODS to be used with DaaSes, we need to extract DaaSes related description and represent them in DEMODS. This can be done through the engineering process of DaaSes and during the operation of DaaSes. In particular, for each type of representation, it is important to have a corresponding parser to extract different pieces of service and data information such as service description, data asset description, or cost model. For example, with respect to the use of WSDL with annotations for DaaS description, an XML parser is needed to extract information about the service. On the other hand, if the description is done through annotations in HTML documents, an HTML parser is needed. Despite the difference in implementation, these parsers must provide a set of unified operations to extract service and data description. In this way, assume that a user is only interested in data categories of published data assets, he can call the corresponding operation from parsers to extract this information without worrying about the representation of DEMODS. All parsers could be incorporated into different manual and automatic description extraction and collection processes as follows.

- Separate tools/processes that include parsers can be used by DaaSes themselves, service providers and data providers to upload service and data descriptions. These tools will extract the right information and store into *DEMODS Information Services* which manage DEMODS-based information. These tools/processes can be provided as a stand-alone one used manual by humans as well as can be integrated into DaaSes so that when the DaaSes have new descriptions or new data assets added into the DaaSes, the DaaSes automatically send descriptions to DEMODS Information Services. A popular way, for example, is that these tools/processes can be integrated into existing interfaces for data providers in contemporary DaaSes. In this model, DEMODS information could be collected from DaaS providers by “push” actions: DaaS providers are expected to actively submit their service and data description to the service.
- Crawlers that include parsers can automatically monitor and extract information for DEMODS Information Services. This model works in a similar way to the well-known web crawler model but for *DEMODS Information Services*. In this model, DEMODS information is collected with “pull” actions: a crawler is designed to visit web pages of DaaS providers to collect service

information, which can be either in the description file or in metadata of HTML documents, e.g., according to hRESTS model.

As we discussed above, we support the use of *DEMODS Information Services* to store DEMODS information. Such services can be an element in a specific data marketplace or they can serve for different data marketplaces. From these services, one can find out DaaSes and relevant data assets. However, we can also link WSDL-based and REST-based service description of DaaSes, usually obtained directly from the application server hosting DaaSes, to corresponding external DEMODS. For example, it is possible to create annotations for the service description as in SAWSDL [5] by employing the "attrExtensions" elements of in the SAWSDL to link DEMODS information. Another way is that links to DEMODS description can be embedded directly in HTML documents by exploiting hRESTS model [27].

The two above-mentioned models of extracting description could be applied in the development of new DaaSes or in the integration of existing DaaSes. In particular, it would be easier for DaaS providers to use DEMODS to provide description information when engineering new DaaSes. Relying on DEMODS, DaaSes could be responsible for providing tools for collecting relevant types of descriptions for data assets and perform syntax and semantic compliance/validation of collected descriptions, while data providers have to actually provide descriptions about their data assets when the data providers offer the data assets via DaaSes. This way would avoid the situation that DaaSes have to collect all descriptions, which, in many cases, is too costly for DaaSes. Furthermore, this way fits well with the sharing liability and responsibility between DaaSes and data providers when offering data assets in data marketplaces.

For existing DaaSes, depending on whether it is easy to modify the existing HTML service description or to collect data to build a service description file, the proper approach could be selected. Once the description is done, the next step could be to contact centralized servers to submit the description file or the link to the HTML description document. Alternative, a plugin could be installed locally to point to the description file or the HTML document. This plugin will then take care of the information submission process and send updated information to the server at interval time or when the description file is changed.

VI. Prototype and Experiment

A. Prototype

We have developed a prototype of DEMODS and *DEMODS Information Service*. Our DEMODS information is currently based on XML that provides mechanisms to link to different external information models based on XML, OWL and RDF. The service was implemented on top of Google Apps Engine (GAE) [1] to provide features for user to upload, search, and retrieve DEMODS-based description. In this prototype, we developed proof-of-concept models for *General Service Description, API Description* and *Data Asset Description* (see Figure 2). For *External Information Models*, we utilized ODATA [14] for domain definition and the Abstract Data Contract Model [23] for data contract. For QoS and quality of data we reuse our previous work in [26] while we used a simple category classification description based on tagging. Due to the lack of existing information models that cover suitable pricing descriptions for DaaSes, we have also developed a model for describing pricing information that covers different payment plans as follows.

- *Payment on access* (API call): consumers are charged every time they call provided APIs in the service to retrieve data. To describe this pricing model, the API usage fee has to be included in the API description.
- *Payment on resource consumption*: consumers are charged on the amount of resource consumption to process the request and return the data to users. For example, Amazon charges the resource consumption based on I/O requests to data assets stored in its EBS. In this case, the pricing description can include only basic unit prices (e.g., how much does it cost for one processing hour) but the real cost of DaaS usage can only be determined at runtime.
- *Payment on data type and data size*: users are charged on the type and the size of the requested data. An example of this model is the pricing model of Gnip [4]. Similar to the previous case, only basic unit prices can be described.
- *Payment on plan* (fixed payment in a period): consumers subscribe for data usage in a period (e.g., a week, a month, or a year) and only pay once for this period with or without maximum limitations for how frequent they access data and how much data they retrieve from DaaSes. This model is well-supported in current data marketplaces.

As illustrated in Figure 3, without considering GUI, the prototype consists of three main components: the parsers, the search engine, and the database. Supporting for these components is a crawler, which is able to follow external information models to extract information from outside DEMODS.

- The parser is in charge of extracting DEMODS information and putting the extracted information in the database. There could be multiple parsers in this component including XML parser to parse XML based description, HTML parser to parse HTML documents, etc. In the current prototype, however, we only implemented an XML parser.
- The search engine is responsible for getting DEMODS information from the database to serve query purpose. Similar to the parser component, there could be multiple sub-components inside the search engine, which supports different search levels from simple to advanced. In this prototype, we supported simple search features that allows to search data assets based on different criteria using

[1] The service is accessible at http://demodsmanagement.appspot.com and supporting materials are available at http://www.infosys.tuwien.ac.at/prototype/SOD1/demods.html

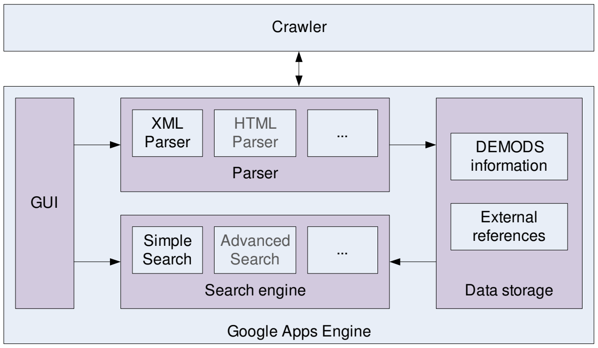


Fig. 3. A prototype of DEMODS Information Service

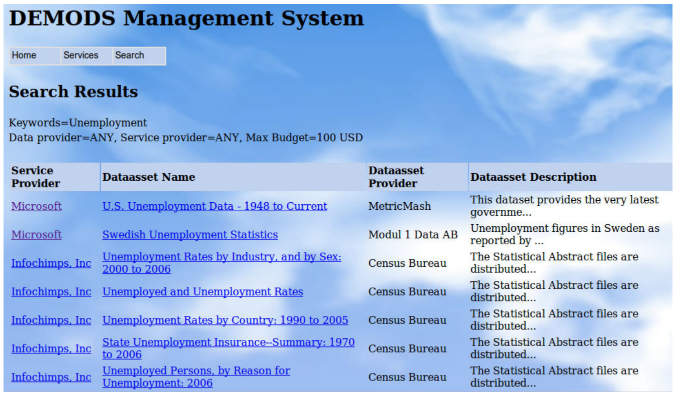
underlying GAE query mechanisms.

- Finally, the DEMODS database consists of two main parts: one is used to store DEMODS information while the other is used to store links and information obtained from external information models.

B. Experiments

1) Using DEMODS for existing DaaSes: We conducted case studies to see how DEMODS can be used for existing DaaSes. It is interesting to realize that Infochimps offer two types of DaaS: one service provides downloadable data assets (obtained as a whole) while the other supports data assets with APIs for data extraction. On the other hand, Microsoft Azure Marketplace only supports data assets with APIs while Amazon Public Datasets only provides data assets to download. These DaaSes employ a very simple category to classify data assets in which only Microsoft Azure Marketplace specifies the structure of data assets using OData. The other two DaaSes provide no description on data asset structure. This simple type of description is well imported to DEMODS. Actually, DEMODS is able to provide better description using category classification, and domain definitions imported from outside. In terms of pricing model, these DaaSes define only few pricing options and apply these options to all data assets and APIs. While Microsoft Azure Data Marketplace and Infochimps also define data contract/license, there is no such definition in Amazon Public Datasets. Again, since DEMODS is able to carry much more detail information, these descriptions could be easily imported to DEMODS. In general, while DEMODS could be used for all existing DaaSes, it provides a high level of flexibility and extensibility to all future DaaSes. Table II summarizes our mappings of key descriptions of existing DaaSes into DEMODS. In overall, our DEMODS is flexible and extensible, while description models employed in these studied DaaSes are not.

2) Examples of information search: Given information from the above three DaaSes, we illustrate a simple query, which we cannot get a satisfied answer from any existing Internet search engine or keyword search/listing features provided by existing DaaSes, to prove the usefulness of DEMODS and DEMODS Information Service. The query is to find data assets having statistics of unemployment. If we execute this query with Internet search engines such as Google, we will receive a bundle of results none of which is useful. However, when executing this query leveraging DEMODS information, for example by specifying "Unemployment" as the search keyword, DEMODS will present useful results about data assets satisfying the query. In particular, there are a number of data assets provided by Infochimps that satisfy the search keyword while there are only two data assets provided Microsoft Azure Marketplace and no data asset provided by Amazon Public Datasets. The search can also be further refined by, for example, specifying the maximum budget (e.g., to 100 USD) to retrieve the desirable data asset and obtain the filtered result as shown in Figure 4. Note that while we only implemented a simple search engine that supports search over four criteria: data asset description, data asset provider, service provider, and cost in our prototype, advanced search features can be plugged into our *DEMODS Information Service* to support search conditions that cover different aspects, such as data agreement, pricing models, quality of data, etc.



DEMODS Management System

Search Results

Keywords=Unemployment
Data provider=ANY, Service provider=ANY, Max Budget=100 USD

Service Provider	Dataasset Name	Dataasset Provider	Dataasset Description
Microsoft	U.S. Unemployment Data - 1948 to Current	MetricMash	This dataset provides the very latest governme...
Microsoft	Swedish Unemployment Statistics	Modul 1 Data AB	Unemployment figures in Sweden as reported by ...
Infochimps, Inc	Unemployment Rates by Industry, and by Sex: 2000 to 2006	Census Bureau	The Statistical Abstract files are distributed...
Infochimps, Inc	Unemployed and Unemployment Rates	Census Bureau	The Statistical Abstract files are distributed...
Infochimps, Inc	Unemployment Rates by Country: 1990 to 2005	Census Bureau	The Statistical Abstract files are distributed...
Infochimps, Inc	State Unemployment Insurance--Summary: 1970 to 2006	Census Bureau	The Statistical Abstract files are distributed...
Infochimps, Inc	Unemployed Persons, by Reason for Unemployment: 2006	Census Bureau	The Statistical Abstract files are distributed...

Fig. 4. Example of the search result returned by DEMODS Information Service

3) Examples of description collection: When a provider wants to release a new DaaS or when an existing DaaS wants to further expose its service using DEMODS, the provider can leverage our prototype management service to submit its service and data description by simply following these two manual steps. First, the provider obtains the XML schema of DEMODS (as a description template) and prepare necessary description complying with DEMODS. After that, the provider can upload its description file into our prototype. Alternatively, the provider can develop a tool to produce DEMODS-compliant description and upload its description. Note that while the current prototype requires the provider to upload service description manually, it is straightforward to provide Web services APIs for collecting DEMODS information, including an HTML parser that extracts information from HTML file to support description in HTML and plugins that allow to extract information directly from description files inside DaaSes in other to allow information to be updated easier.

VII. Conclusions and Future Work

While the DaaS model has been increasingly used for facilitating the sharing and integration of data assets via open APIs, the description of DaaSes and their data assets have

Description	Microsoft Azure Marketplace	Infochimps	Amazon Public Datasets	DEMODS
Data asset UUID	public URI	public URI	ids of EBS snapshot	mapped to uuid of data assets
Category	simple hierarchical category	simple hierarchical category	simple hierarchical category	mapped to tags in the category classification schema
Pricing	free, subscription model with limited transactions for individual data assets	free, subscription pricing models for all APIs (at service level) and premium APIs pricing models for individual data assets	subscription pricing models and pay-per-use models for all public data assets	mapped to pricing models in General Service Description
Downloadable data assets	not supported	size, format, licensing, quality, price	size, snapshot ID, contract	mapped to data asset information, including size, format, data contract, quality of data, and pricing
Data assets with APIs	detailed APIs, data licensing, and pricing models	detailed APIs and pricing models	does not exist	mapped to data asset APIs, pricing and contract models
Data structure	annotated with OData	no	no	mapped to OData

TABLE II
MAPPING DESCRIPTION FROM REAL-WORLD DAASES TO DEMODS

not been well-researched. In this paper, we have introduced a novel conceptual information model for DaaSes. Our DEMODS is capable of linking different types of descriptions that characterize both service and data aspects found in DaaSes.

Currently, we have built a prototype of DEMODS to test our proposed model with well-known DaaSes. Our next step is to integrate this prototype with advanced data and service discovery features for DaaSes. After that, we will investigate dynamic data integration and query optimization based on automatic discovery of data from DaaSes. Furthermore, we aim at integrating our DEMODS into a specific DaaS for facility monitoring data marketplaces. Finally, a thorough performance evaluation will be executed.

ACKNOWLEDGEMENTS

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

REFERENCES

[1] *Amazon Data Sets*, http://aws.amazon.com/publicdatasets/.

[2] *Microsoft Data Market*, https://datamarket.azure.com/.

[3] *Infochimps*, http://www.infochimps.com/.

[4] *Gnip*, http://gnip.com/.

[5] *Semantic Annotations for WSDL and XML Schema*, http://www.w3.org/TR/sawsdl/.

[6] S. A. McIlraith, T. C. Son, and H. Zeng, "Semantic web services," *Intelligent Systems*, vol. 16, no. 2, pp. 46–53, 2001.

[7] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara. (2004) Owl-s: Semantic markup for web services. [Online]. Available: http://www.w3.org/Submission/OWL-S/c

[8] M. Treiber, H. L. Truong, and S. Dustdar, "Semf - service evolution management framework," in *EUROMICRO-SEAA*. IEEE, 2008, pp. 329–336.

[9] J. Cardoso, A. P. Barros, N. May, and U. Kylau, "Towards a unified service description language for the internet of services: Requirements and first developments," in *IEEE SCC*. IEEE Computer Society, 2010, pp. 602–609.

[10] G. Singh, S. Bharathi, A. Chervenak, E. Deelman, C. Kesselman, M. Manohar, S. Patil, and L. Pearlman, "A metadata catalog service for data intensive applications," in *Proceedings of the 2003 ACM/IEEE conference on Supercomputing*, ser. SC '03. New York, NY, USA: ACM, 2003, pp. 33–. [Online]. Available: http://doi.acm.org/10.1145/1048935.1050184

[11] V. Khatri and C. V. Brown, "Designing data governance," *Commun. ACM*, vol. 53, pp. 148–152, January 2010. [Online]. Available: http://doi.acm.org/10.1145/1629175.1629210

[12] A. Keller and H. Ludwig, "The wsla framework: Specifying and monitoring service level agreements for web services," *J. Network Syst. Manage.*, vol. 11, no. 1, pp. 57–81, 2003.

[13] G. R. Gangadharan and V. D'Andrea, "Service licensing: conceptualization, formalization, and expression," *Service Oriented Computing and Applications*, vol. 5, no. 1, pp. 37–59, 2011.

[14] *The Open Data Protocol (OData)*, http://www.odata.org/.

[15] *Dublin Core Metadata Initiative*, http://dublincore.org/.

[16] *Socrata*, http://www.socrata.com/.

[17] I. V. Papaioannou, D. T. Tsesmetzis, I. Roussaki, and M. E. Anagnostou, "A qos ontology language for web-services," in *AINA (1)*. IEEE Computer Society, 2006, pp. 101–106.

[18] G. Dobson, R. Lock, and I. Sommerville, "Qosont: a qos ontology for service-centric systems," in *EUROMICRO-SEAA*. IEEE Computer Society, 2005, pp. 80–87.

[19] "Snap ontologies – simple news and press ontologies," http://data.press.net/ontology/, last access: 6 Oct, 2011.

[20] Y. Ding, E. K. Jacob, M. Fried, I. Toma, E. Yan, S. Foo, and S. Milojević, "Upper tag ontology for integrating social tagging data," *J. Am. Soc. Inf. Sci. Technol.*, vol. 61, pp. 505–521, March 2010. [Online]. Available: http://dx.doi.org/10.1002/asi.v61:3

[21] S. Scerri, M. Sintek, L. van Elst, and S. Handschuh, "Nepomuk annotation ontology specification," http://www.semanticdesktop.org/ontologies/nao, last access: 6 Oct, 2011.

[22] "Open annotation collaboration," http://www.openannotation.org/, last access: 6 Oct, 2011.

[23] H.-L. Truong, G. Gangadharan, M. Comerio, S. Dustdar, and F. D. Paoli, "On Analyzing and Developing Data Contracts in Cloud-based Data Marketplaces," in *Proceedings of the Asia-Pacific Services Computing Conference 2011 (APSCC 2011)*, 2011, to appear.

[24] A. S. Aparício, O. L. M. Farias, and N. dos Santos, "Applying ontologies in the integration of heterogeneous relational databases," in *Proceedings of the 2005 Australasian Ontology Workshop*, 2005, pp. 11–16.

[25] D. Dou and P. LePendu, "Ontology-based integration for relational databases," in *Proceedings of the 2006 ACM symposium on Applied computing*, 2006, pp. 461–466.

[26] 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.

[27] J. Kopecky, K. Gomadam, and T. Vitvar, "hRESTS: An HTML microformat for describing RESTful Web services," in *Proceedings of the International Conference on Web Intelligence and Intelligent Agent Technology*, 2008.