GFD-I.163
OGF DAIS Working Group

Mario Antonioletti, EPCC
Carlos Buil Aranda, UPM
Oscar Corcho, UPM
Miguel Esteban-Gutiérrez, UPM
Asunción Gómez-Pérez, UPM
Isao Kojima (Corresponding Author), AIST
Steven Lynden, AIST
Said Mirza Pahlevi, AIST

Category: INFORMATIONAL

December 30, 2009

WS-DAI RDF(S) Realization: Introduction, Motivational Use Cases and Terminologies

Status of This Memo

This document provides information about the initiative for the provisioning of access to RDF(S) data resources by means of specific realizations of the WS-DAI Core specification.

Copyright Notice

Abstract

The Database Access and Integration Services Working Group (DAIS-WG) has submitted three specifications to the Open Grid Forum (OGF) recommendation track [WS-DAI, WS-DAIR, WS-DAIX]. These specifications define a basic set of interfaces, properties and patterns for service-based access to data. The core WS-DAI specification outlines a set of generic interfaces and properties that are common to most types of data access. These may then be extended to access specific types of data. For instance, the WS-DAIR and WS-DAIX specifications extend the base specification to provide access to relational and XML types of data respectively.

This document outlines and motivates a further extension to the WS-DAI family of specifications to provide access to RDF(S) data. This will define a standard mechanism for accessing RDF(S) data in a manner consistent with the framework defined by the WS-DAI core specification. The main outcome of this work will be two specifications that provide complementary ways for accessing RDF(S) data: by using the W3C defined SPARQL [SPARQL] query language or through the use of ontological primitives.

This document motivates this work by presenting an overview of the role of RDF(S) in a grid context with several motivational use cases.

Table of Contents

1. Introduction

Grid technologies aim to provide the framework to enable the dynamic, flexible sharing of computational, data and other types of resources through interoperable middleware based on open standards. To successfully achieve this end one must be able to unambiguously interpret metadata about resources in order to be able to discover, utilise correctly and effectively combine resources together, usually in a dynamic manner, to solve problems.

> The Semantic Web community [SW] is currently leading research and development work in the area of semantic technologies, with a main objective being the provision of a "common framework that allows data to be shared and reused across application, enterprise, and community boundaries" [http://www.w3.org/2001/sw/], building on the data model defined by the Resource Description Framework specifications [RDF-XML, RDFS, RDF-SEMANTICS], also known as RDF(S). In terms of its adoption, RDF(S) is being used extensively to represent large amounts of data by a number of applications worldwide. For example, the UniProt [www.uniprot.org] Protein Database contains 262 million RDF triples, DBpedia [http://wiki.dbpedia.org/] contains over 270 million RDF facts and the Linking Open Data project [http://linkeddata.org/] now provides 4.7 billion RDF triples in total. In the same way that RDF(S) is a fundamental building block of the Semantic Web, it naturally follows that RDF(S) data resources are a key element for metadata exposure and provisioning.

This document introduces and motivates the definition of a set of service-based interfaces for accessing RDF(S) [DAIRDFS], based on the OGF WS-DAI specification for data access and integration together with a set of use cases highlighting potential scenarios to which this technology could be applied in the context of a regular web service environments or as part of a grid fabric.

2. Motivation

The next generation of semantically aware grid technologies need to be able to provide metadata to support the virtualisation of distributed computation, storage, and communication over a large number of resources [GRID]. This is challenging when systems are loosely coupled and heterogeneous, where any grid node may provide, at any point in time, new services, functions, or, in general, new resources that are unknown a priori to its clients or the other grid nodes. In order to incorporate these new elements into other applications or middleware, or to cooperate with them, not only do they have to be made available and accessible in a standardized way, but also visible and adequately described. Metadata plays a crucial role for this to be achievable; however [S-OGSA] identifies a number of reasons as to why metadata becomes difficult to interpret in existing grids, including: "knowledge burial", the tendency for resource metadata to be buried in middleware code, libraries, different database schemas and XML documents. One way of mitigating this issue is through the use of vocabularies that are defined, agreed and shared by a community, thus ensuring a degree of interoperability across applications and/or middleware that exploit this metadata. Examples of resource description vocabularies are: GLUE [GLUE], the forthcoming Network Mark-up Language (NML) currently being developed by the OGF NML working group [http://forge.gridforum.org/sf/projects/nml-wg] and the DMTF Common Information Model (CIM) [http://www.dmtf.org/standards/cim/]. Through the use of these vocabularies, communities can tackle challenges like: resource discovery and selection (also known as matchmaking), brokering, monitoring, accounting, etc.

These vocabularies have been traditionally defined using XML Schema, which dictates both the structure to be used for resource descriptions as well as the set of data types needed for such structures. Hence, resource descriptions are expressed as XML documents that follow the corresponding schema. This approach is good enough for closed environments where the types of resources, or the information that can be described, are known a priori. However, this approach is too rigid for open environments where new elements are incorporated dynamically.

The use of languages like RDF(S) offer more flexibility in the description of metadata. The work described in this document aims to support this effort by extending the WS-DAI specifications, which already provide web service access to XML and relational data, to also encompass RDF(S) data. Standard programming languages APIs for accessing databases, e.g. JDBC, have been widely used to save on programming effort and promote interoperability; the WS-DAI

specifications aim to bring the same benefits to service-based computing. The WS-DAI specifications provide a set of WSDL [WSDL] defined interfaces for managing, querying and describing various properties associated with data resources, i.e. the DBMS that manage the data. Thus an important aspect of using this approach is that the interfaces defined in the WS-DAI specifications can be combined with those defined by other web service standards that concentrate on other areas, for example, security. In addition to this, the web service interfaces provide a programming language independent way to access the underlying data resources. This is of particular value to clients accessing RDF stores which do not provide consistent APIs for accessing the RDF data. Thus the abstraction layer provided by the proposed RDF DAIS interfaces will allow clients to contact the underlying RDF stores in a consistent manner regardless of what the underlying storage engine is. In addition, although the SPARQL query language already has an associated W3C recommendation web service-based protocol [SPROT] for executing SPARQL queries.

The RDF(S)-based WS-DAI specifications motivated in this document are required for the following reasons:

- The WS-DAI specifications set out standard patterns for interacting with data resources within the context of service-based computing. For example, operations and properties exist for exposing information about a data resource's ability to support various features such as transactions and concurrency. Furthermore, WS-DAI can leverage off existing Web Service specifications such as WS-ResourceProperties [WS-ResourceProperties] for exposing resource properties and WS-ResourceLifetime [WS-ResourceLifetime] for resource lifetime management, both of which form part of the Web Services Resource Framework [WSRF]. In contrast, the SPARQL Protocol defines a single query operation and associated fault messages but lacks the range of operations, properties and faults defined within the WS-DAI specifications in order to fully support access to data resources in a service-based setting. An RDF(S) realization of WS-DAI is therefore required to provide this support in an RDF(S) setting.
- The SPARQL Protocol's query operation is analogous to the direct data access query pattern specified by WS-DAI, where the entire dataset formed as the result of a query is returned to the client within a response message. The WS-DAI specification defines a second pattern, indirect data access, where the result of a query is made available as a new data resource, i.e. implementing the factory pattern. This pattern supports an indirect form of third-party delivery, can be used to avoid unnecessary data movement, and allows a client to pull data from a data resource rather than have it returned all at once in a single response message. This access pattern is important in a wide range of scenarios including distributed query processing and providing scalable/reliable data access.
- The execution of queries is not the only means by which a client may wish to interact with an RDF(S) data resource. The interfaces provided by RDF(S) storage systems, for example the Jena Semantic Web Framework [http://jena.sourceforge.net/] Ontology API, provide a range of mechanisms for directly manipulating an ontology. As such APIs vary depending on the specific storage system used, an application developer must change to a different API if some resources in different organizations are stored using different systems such as Oracle RDF or Sesame [http://www.openrdf.org/]. The greater the number of storage systems that are used, the more APIs that need to be known the greater the effort required to build an application. The WS-DAI-RDF(S) Ontology specification aims to provide a standardized set of ontology handling primitives for interacting with RDF(S) data resources, hiding any syntactic and platform-dependent aspects, enabling their use in an open, interoperable way.

Currently, the W3C has published several specifications to access RDF(S) data, such as the access protocol [SPROT] and query results format [RESULTS] for the SPARQL [SPARQL] query language. However, it is important to note that this set of W3C specifications are targeted only at extracting data from RDF(S) repositories. That is, they currently do not define a means for creating, updating or deleting RDF(S) data, although particular proposals exist for doing so (see [SUPDATE]).

These factors motivate our work on the provision of a WS-DAI-based standard supporting RDF(S) data resources. The goal is to develop a single framework that satisfies the

requirements outlined above, including scalable data access patterns and a layer of abstraction allowing for consistent interaction with underlying RDF management systems.

3. Specification Overview

3.1. Specification Organization

We have identified two different ways of interacting with RDF(S) resources: firstly we follow a query approach, in which a client can retrieve the contents of a resource using SPARQL queries; for the second we follow an ontological approach that enables a client to explore and modify a resource using a set of primitives for accessing ontologies. In addition to the fact that they both share a common purpose in supporting access to RDF(S), the common denominator to both approaches is the type of underlying (RDF) data resource and the (RDF) data model managed by the data resource. Thus, two specifications are proposed as new WS-DAI realizations for accessing RDF(S) data:

1) WS-DAI RDF(S) Querying [WS-DAI-RDF(S)-Query]: this specification provides a query language interface to RDF data. This is based on the set of W3C SPARQL [SPARQL] and supports several extensions including the indirect access pattern mandated by the WS-DAI core specification.

2) WS-DAI RDF(S) Ontology [WS-DAI-RDF(S)-Ont]: this specification provides an API style ontology handling set of primitives based on the RDF(S) model. These primitives provide various operations including updates to the ontology.

These approaches are not mutually exclusive, as using one of them does not imply that the other one cannot be used at the same time on the same resource. Furthermore, they are complementary, as they offer different mechanisms for interacting with data resources and each of them is targeted to fulfil different specific access requirements.

As each of the two approaches provides a different kind of interface to RDF(S) resources, they need to be addressed by different specifications. Nevertheless, the specifications should not be totally decoupled, as they share a common purpose, data model and principal actor: the RDF(S) data resource.

Figure 1 shows these specifications and their relations to existing set of WS-DAI specifications. Both the RDF(S) Ontology and Query access specifications are WS-DAI realizations, like the WS-DAIR [WS-DAIR] specification for relational data and the WS-DAIX [WS-DAIX] specification for XML data, but they have a common purpose, supporting access to RDF(S) data resources, the motivation for which is presented in this document.

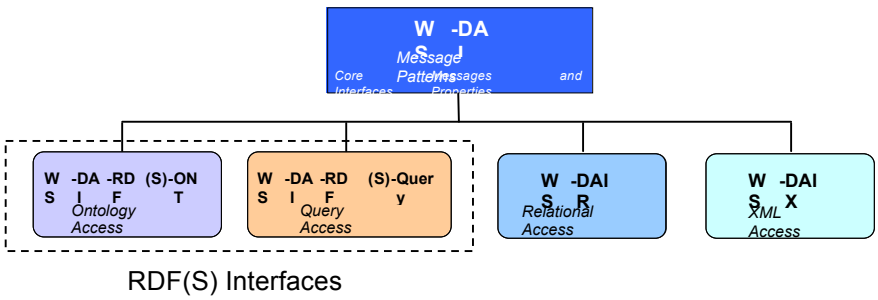

Figure 1: The WS-DAI family of specifications

3.2. Terminologies

As the two RDF(S) related specifications share a common purpose, the set of interfaces these specifications provide can be conceptually grouped together, as illustrated in Figure 1, to form the set of WS-DAI-based interfaces supporting RDF, defined as follows:

RDF(S) Interfaces: The base interfaces and corresponding properties defined in the WS-DAI specification extended to provide access to RDF(S) data resources.

The WS-DAI specification family is based on the concept of a data resource. Relational and XML data resources are defined in the WS-DAIR and WS-DAIX specifications, respectively. For the WS-DAI RDF(S), we have defined RDF(S) Data Resource as follows:

RDF(S) Data Resource: A data source or sink that is based on the RDF data model, together with any associated management infrastructure that exhibits capabilities that are characteristic of RDF repositories. The management infrastructure may also exhibit RDF(S) model based views, exposing RDF Schema entailment capabilities over the resource. An RDF(S) Data Resource is illustrated in Figure 2.

As described in Section 2, two specifications aim to provide different views for the same RDF data. An RDF(S) Data Resource can be handled as a set of RDF triples [RDF-CONCEPTS] (instances) or an ontological hierarchy which is based on the RDF(S) model [RDF-CONCEPTS].

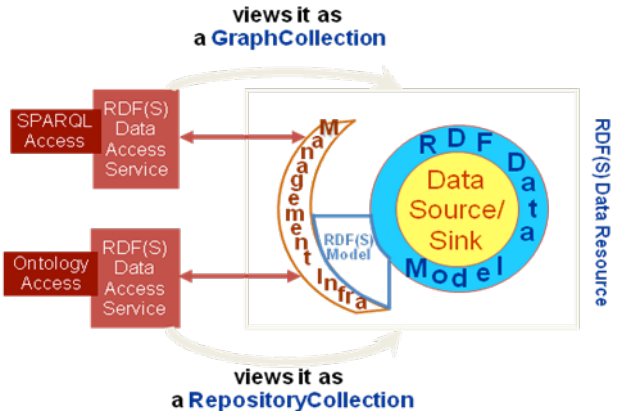

Figure 2: An RDF(S) Data Resource

This means that there are two different ways in which the data can be viewed. For instance, a set of RDF triples can be handled as an RDF Graph from the instance point of view. Since the term Graph is a defined term in [RDF-CONCEPTS], these triples must be represented as a Graph when using the WS-DAI-RDF(S) Querying specification. On the other hand, the RDF(S) Ontology specification presents a view that is based on an ontology hierarchy that can be manipulated by ontological primitives, which are defined as follows:

Ontological access primitive: A data access operation based on the model/formalism used for representing the data, which takes into account the structures defined by the formalism and the relationships between them.

Ontological access primitives are performed on a Repository, which is defined as follows:

Repository: A set of RDF triples that are defined together. This term is synonym of the term RDF Graph as defined in [RDF-CONCEPTS].

The specifications also support operations on collections of RDF(S) data resources, in which case the RDF(S) Querying specification presents a GraphCollection view and the RDF(S) Ontology specification presents a RepositoryCollection view, as defined below.

GraphCollection: A set of RDF graphs.

Repository Collection: An entity that manages sets of repositories.

Table 1 shows how the above terms fit into the views provided by the two specifications.

	WS-DAI RDF(S) Ontology	WS-DAI RDF(S) Querying
An RDF(S) data resource (see Fig 2)	A Repository Data Resource	An RDF Graph Data Resource
A set of RDF(S) data resources	A RepositoryCollection Data Resource	A GraphCollection Data Resource

Table 1: Term relationships between some terms of two specifications

3.3. WS-DAI-RDF(S) Querying

The objective of the querying specification is to provide a set of set-oriented declarative access methods to execute queries submitted by a client. This specification does not specify its own language to access the RDF(S) data resources. Instead, it acts as a channel for RDF queries to be conveyed to the appropriate data resources. For instance, for RDF(S) data resources, or for data resources that supports RDF type queries, the query language supported is SPARQL, a W3C recommendation.

SPARQL has four query forms: CONSTRUCT, DESCRIBE, SELECT, and ASK. The first and second forms return an RDF graph as the result of a query (CONSTRUCT returns an RDF graph constructed by substituting variables in the query patterns, while DESCRIBE returns an

RDF graph that describes the resources found). In contrast to these two forms, the results of the other two are not RDF graphs: SELECT returns all, or a subset of, the variables found in a query pattern match; ASK returns a boolean value indicating whether there was a match for a query pattern.

In addition to the SPARQL query language, the W3C has recommended the following related standard specifications to access remote/distributed RDF data using SPARQL:

- SPARQL Query Results XML Format [RESULTS]: is an XML format for variable bindings and boolean results defined by the SPARQL query language.
- SPARQL Protocol for RDF [SPROT]: is a protocol for conveying SPARQL queries from query clients to SPARQL query processors.

The approach taken in the WS-DAI-RDF(S) Querying specification is to keep as much compatibility with the existing W3C standards as possible while satisfying the WS-DAI core specification at the same time, in particular by fully supporting the SPARQL query language and its associated XML result format.

Direct Access and Indirect Access

One of the key features of the WS-DAI specification, illustrated in Figure 3 within the context of WS-DAI-RDF(S), is that of direct/indirect access to data resources. Direct access allows the results of a request to be delivered to a consumer directly in the response message. This is one of the two access patterns which the WS-DAI core model describes. To cater for this mode of operation the specification defines an interface, SPARQLAccess, for accessing an RDF(S) data resource using SPARQL.

Indirect access is the other access pattern which the WS-DAI core model supports. This allows data, usually the result of a query, to be accessed by means of a new service-managed data resource, and thus data is not returned directly to the consumer. Indirect access can be very useful when it is anticipated that the size of a query result will be large.

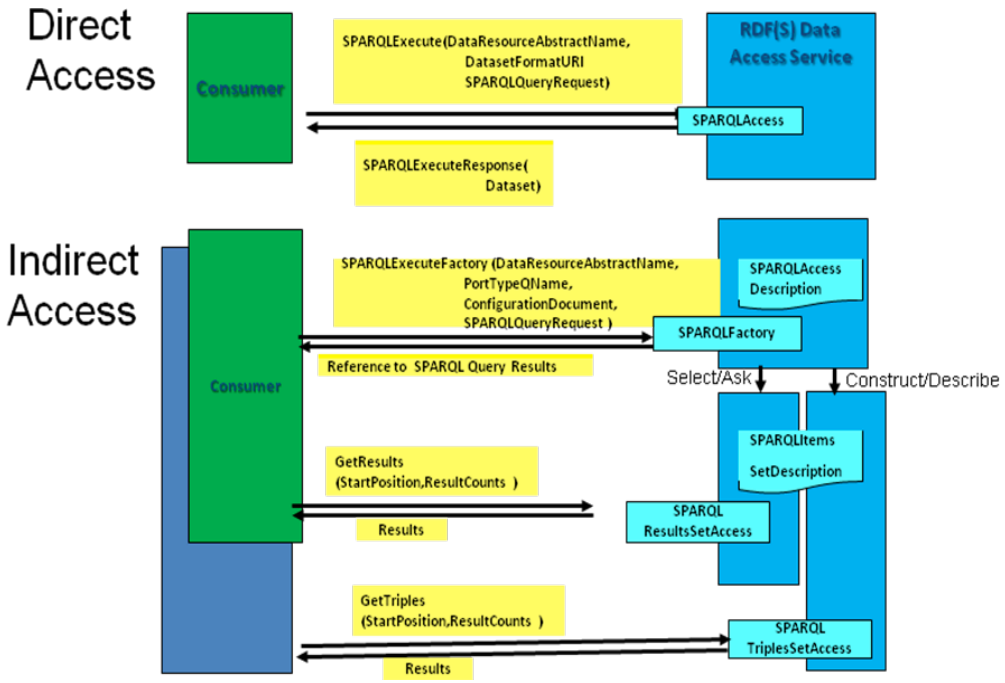

Figure 3: Overview of WS-DAI RDF(S) Querying Specification

In order to access query results derived through indirect access, two interfaces have been defined that provide specialized access to these results: TriplesSetAccess and ResultsSetAccess. These interfaces may be made available through different data access services, the end points being returned to the client as a result of one of the indirect (factory) operations.

3.4. WS-DAI-RDF(S) Ontology

The objective of the WS-DAI-RDF(S) Ontology access specification is to provide an integral access mechanism for RDF(S) sources that goes beyond the retrieval capabilities offered by the querying specification, whilst providing a simple but complete set of functionalities that abstracts the most general necessities a client may have when working with RDF(S) data sources. Thus, the specification details a set of ontology handling primitives for dealing with the RDF(S) model, hiding the syntactic aspects of RDF(S) and transparently exploiting its semantics.

Data Resources and Interfaces

The specification differentiates several types of RDF(S) data resources, each of them allowing addressing and accessing of RDF(S) sources at different levels of granularity. These data resources can be divided in two groups: placeholders for built-in RDF(S) classes, and convenience abstractions. The diagram depicted in Figure 4 shows the resources defined and the relationships existing between them using UML notation.

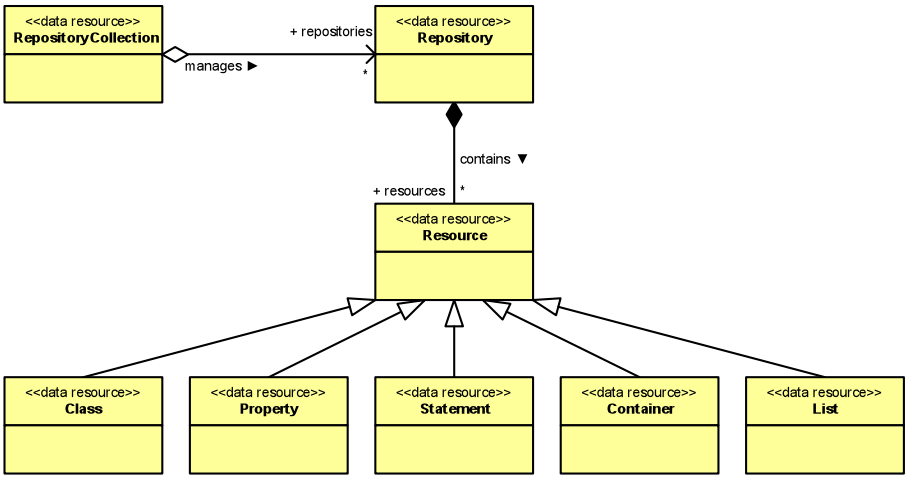

Figure 4: WS-DAI-RDF(S) Ontology Data Resource Model

On the one hand, placeholders for built-in RDF(S) classes (Resource, Class, Property, Statement, Container, and List data resources) provide class-oriented views of an RDF individual (entity or thing). That is, the particular view focus on the specific data that is defined for the RDF individual according to the semantics of the particular RDF(S) built-in class, as defined in [RDF-SEMANTICS].

On the other hand, convenience abstraction data resources (such as RepositoryCollection and Repository) provide a means for dealing with multiple RDF individuals. Thus, a Repository data resource contains data (RDF triples) that simultaneously define multiple RDF individuals. Similarly, a RepositoryCollection data resource aggregates multiple repositories.

Based on this Resource model, direct and indirect access interfaces can be defined. Indirect access uses the factory pattern, which allows for a basic navigation mechanism, based on the creation of new resources to represent data using different interfaces at various hierarchical levels. This allows a a client to browse RDF(S) data resources at different levels of granularity and exploit the semantics of the RDF(S) data represented by RDF(S) data resources via the ontological access primitives supported by the specification.

3.5. Specification Design Policies and Issues

The design of the WS-DAI-RDF(S) Querying specification follows the same approach as the WS-DAIR and WS-DAIX specifications for relational and XML data by extending the core WS-DAI specification with a set of RDF(S) specific properties and operations to support SPARQL querying.

The WS-DAI-RDF(S) Ontology specification is aimed at providing a means for accessing RDF(S) data resources in a comprehensive manner, offering mechanisms for creating, retrieving, updating and deleting contents. The specification defines these mechanisms following the RDF(S) model and semantics, providing clients with different levels of granularity in which to view and use data resources by using different types of data resources and interfaces. As a result, the full specification provides a larger number of interfaces and operations than the other DAIS realizations (the WS-DAIR, WS-DAIX and the WS-DAI-RDF(S) specifications).

From a consumer/service provider point of view, the usefulness of the specification depends on their specific requirements, especially when dealing with RDF(S) data sources, that is: what needs to be done and how the consumer expects to be able to do it. For this reason, in order to facilitate the adoption and implementation of the specification by the community, the WS-DAI-RDF(S) Ontology specification has been divided into three different *profiles*, each including an increasing degree of functionality to enable clients to deal with RDF(S) data resources at a finer grain of detail, while ensuring interoperability among any implementations (see the relationships between profiles in Figure 5). For instance, if a service provider implements Profile 0, the consumer can be sure that ALL interfaces and operations defined in Profile 0 have been implemented. Furthermore, the service provider which supports Profile 1 must support Profile 0.

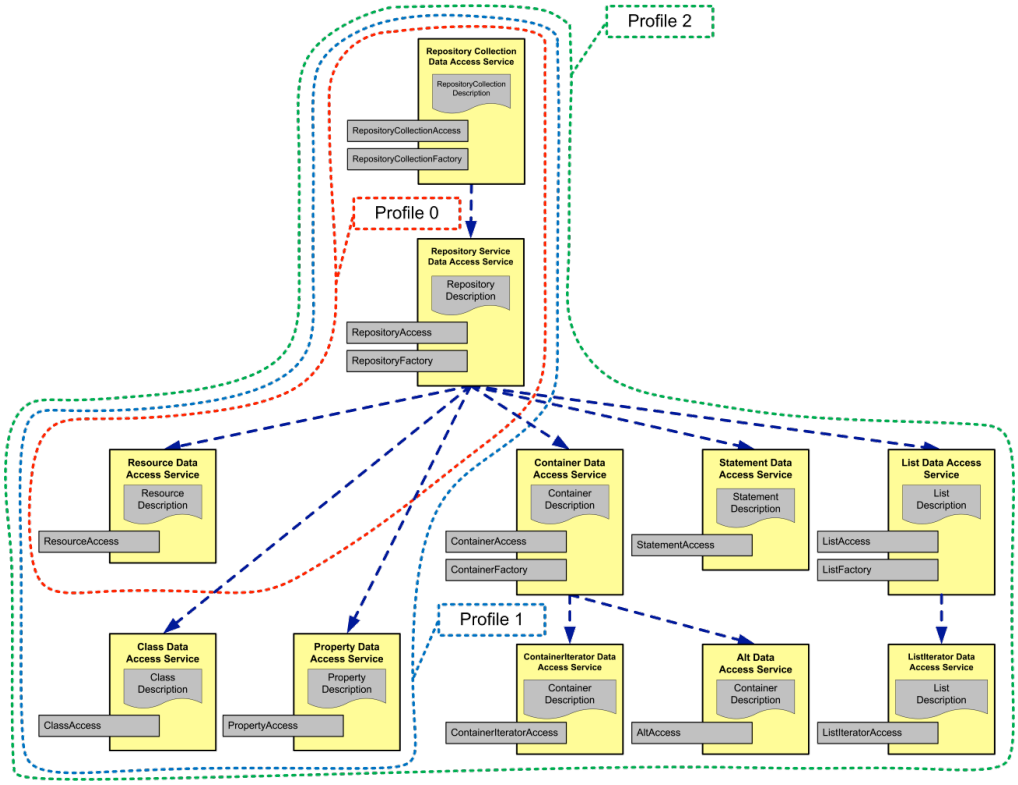

Figure 5: WS-DAI-RDF(S) Ontology profiles

The profiles are thus:

Profile 0: Basic RDF support. This profile includes the minimum set of functionalities needed for accessing RDF data without taking into account the semantics of the RDF(S) model. Within this profile, clients can manipulate the contents of an RDF(S) resource as a whole –using the RepositoryAccess interface– or by directly inspecting the individuals defined within the RDF(S) resource, using the ResourceAccess interface.

Profile 1: RDF Schema support. This profile includes all the functionalities described in Profile 0, enhancing this by taking into account the semantics of the RDF(S) model and by providing additional functionality to work with RDF vocabularies at a conceptual level. Thus, with this profile clients can directly deal with the classes and properties defined within an RDF(S) resource, being able to explore and manipulate their hierarchies and also discover how individuals are classified according to the vocabulary, but without needing to explicitly deal with the underlying syntactic details. Section 4.2 describes a use case that deals with RDF vocabularies to determine the scope of changes detected during resource monitoring, using Profile 1 to provide the means to further exploit the semantics of RDF vocabularies.

Profile 2: Full RDF(S) support. This profile includes all the functionality described in Profile 1, and extends it to deal with the rest of the built-in RDF vocabulary (containers, RDF collections and reifications). This profile provides the means for dealing with additional RDF abstractions using well-known data access patterns, i.e. traversing the members of an RDF collection or a container without requiring any preliminary knowledge about its internal structure using the iterator pattern.

In contrast to the functionality provided by the above profiles, the WS-DAI-RDF(S) Querying specification is designed with the aim of being a minimal extension of WS-DAI, providing support for queries only. For instance, the WS-DAI RDF(S) Querying specification supports SPARQL as a means of interacting with RDF data, which does not yet provide any update functionality. Although SPARQL update languages have been proposed, standards do not yet exist and therefore the WS-DAI-RDF(S) Ontology specification provides the only way in which the WS-DAI-RDF(S) specifications may be used to update RDF data.

4. Motivational Use Cases

This section presents several scenarios that demonstrate the need and usefulness of RDF(S) to describe data and resource metadata as well as motivating the RDF(S) data access methods developed. Other use cases that show the usefulness of RDF(S) data access protocols in different types of applications can be found in [UNCR] .

The first scenario, in Section 4.1, shows how RDF(S) can be used to enable resource matchmaking in a virtual organization, where RDF(S) is used to describe the resources offered by each organization, and how RDF(S) access methods (either programmatic or declarative) facilitate this task.

The second scenario, in Section 4.2, shows how resources in a virtual organization, such as the ones in the aforementioned matchmaking scenario, can be monitored and annotated in order to maintain up-to-date metadata about them, so that future matchmaking tasks can continue to be performed accurately.

The third scenario, in Section 4.3, shows the importance of using a standard access method and the benefits provided by the WS-DAI-RDF(S) Querying specification when processing federated queries over multiple distributed RDF databases. The final scenario, in Section 4.4, shows how the EU funded ADMIRE project is using the WS-DAI-RDF specification to provide access to its registries.

4.1. Grid Resource Matchmaking in Virtual Organizations

Motivation

A grid may include a large number of resources with various intrinsic capabilities distributed across different organizations. The explicit representation of resource metadata, with its adequate exploitation, plays an important role in facilitating effective grid resource discovery and selection, as shown in [TANGDK]. This is a key aspect considered in semantic grid information system architectures and middleware such as S-MDS [S-MDS], S-OGSA [S-OGSA], S-SRB [S-SRB], and the CaBIG [https://cabig.nci.nih.gov/] project's data access services, etc.

Goal

Given a set of repositories and services that store metadata from different types of resources, the goal of a matchmaker is to discover and select appropriate resources for a given task. This can be done by querying the available metadata – either using a high-level RDF query language such as SPARQL or using a specialized data access API – and ordering the matched resources based on specific ordering criteria, i.e. class subsumption relationships.

Requirement Analysis

Each semantic grid information system may collect resource information from different sources in a grid, and maintain the resource metadata using their own proprietary mechanisms. Despite their differences, the metadata representation used by these systems is the same, that is, it is based on the RDF(S) model. Besides, the metadata could be created using the same RDF schema. In this scenario, it is also desirable to retrieve resource metadata from multiple available systems, so that the client may be able to obtain a more complete description of the resources, as the lack of information from one system might be compensated by the information from the others.

Use Case

Figure 6 shows the aforementioned matchmaking scenario implemented using the SPARQL query language. In this scenario, RDF(S) data sources are exposed through RDF(S) data access services which support the WS-DAI-RDF(S) query-based access mechanism. A requester sends a resource request to the matchmaker, specifying the resource requirements as a SPARQL query (1). The matchmaker forwards the query to existing metadata information systems, which also support the same querying capabilities (2, 4, 6 and 8). After receiving the query results (3, 5, 7 and 9), the matchmaker merges the results and forwards them to the consumer (10). Similar work has been proposed and implemented in a semantic web environment [MATCH].

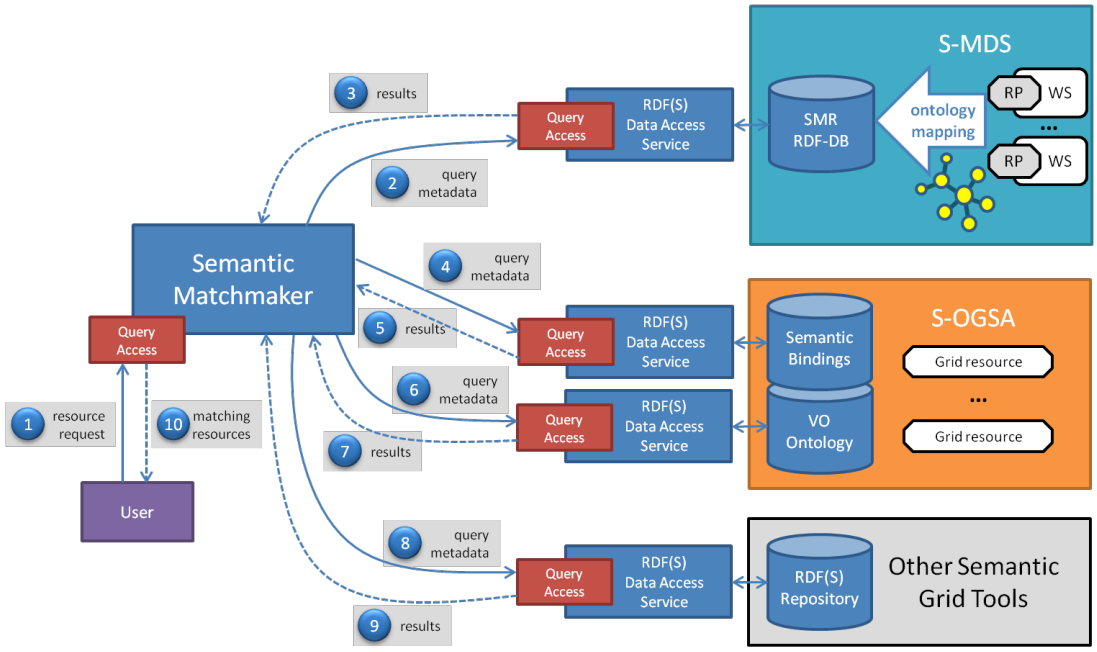

Figure 6: Grid resource matchmaking using WS-DAI-RDF(S) data access mechanisms

4.2. Grid Resource Annotation and Monitoring

Motivation

As previously mentioned, a grid can host a large number of resources with heterogeneous characteristics and capabilities distributed across different organizations, hosting various semantic grid applications and architectures [GRID2SEM] aimed at facilitating the discovery and selection of the resources available by the using metadata for these resources. Providing the means for maintaining valid and up-to-date metadata is fundamental to carry out accurate resource matchmaking for this scenario.

Goal

Consider a scenario in which a set of agents monitor available resources that themselves have monitoring capabilities, i.e. the model developed by the Info Dissemination Working Group [http://forge.gridforum.org/sf/projects/infod-wg] for notifying changes in resource status in a virtual organization. The agents will monitor each resource's characteristics, capabilities, status etc. and compile this information into metadata about each resource, represented using a suitable vocabulary. Thus given a set of such repositories and services that store the metadata and the vocabularies that provide the semantics for the metadata; the goals are to provide a means for creating the metadata using an adequate monitoring vocabulary, and to provide a way of maintaining the metadata stored in the repositories.

Requirement Analysis

The maintenance of this metadata implies browsing, updating and deleting existing metadata already stored in the repositories. Therefore, it is necessary to have a means for both reading and writing the metadata. Furthermore, as both the annotation process and the metadata being managed might be very complex and large in terms of quantity, deleting and generating all the metadata about a resource every time a change occurs may not be feasible. Thus, having fine grained operations for modifying the metadata is worthwhile.

Use Case

Figure 7 schematically depicts this monitoring and annotation scenario. RDF(S) data access services provide a standard access method for the RDF(S) data resources (metadata repositories and vocabulary repositories). Support for updates to the repositories are required, therefore the WS-DAI-RDF(S) Ontology specifications are used. Monitoring agents connect to the resources' monitoring facilities (1). When a change in the resource is detected an agent is

notified(2), the agent browses the vocabulary repositories to check which elements are affected by the specific change (elements that are obsolete, elements that may be out of date, and new elements that may also have to be added) (3). After determining the set of changes that have to be made in the metadata repositories, the agent deletes the obsolete parts of the affected metadata (4), updates those parts that are out-of-date (5), and creates any new part that is required (6).

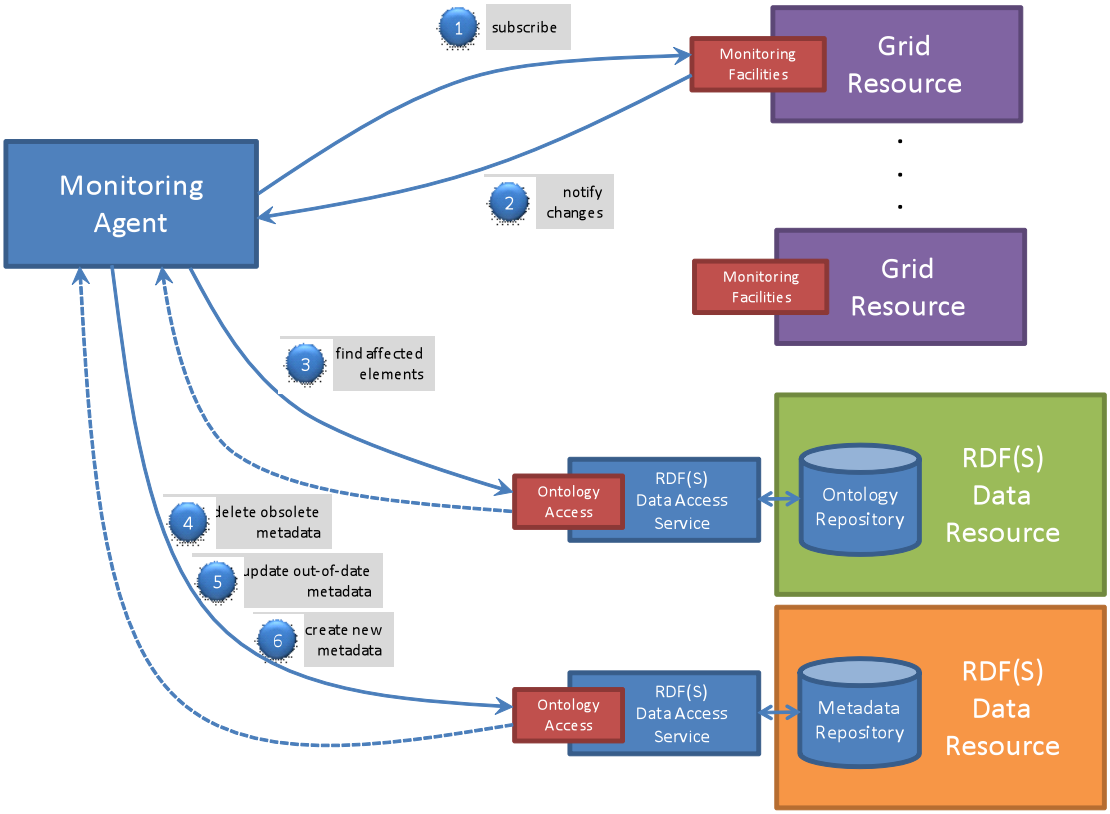

Figure 7: Grid resource monitoring and annotation using WS-DAI-RDF(S) data access mechanisms

4.3. Federated SPARQL (Distributed RDF Data Integration)

Motivation

The distributed and data-intensive nature of service-based grids means that integrating data from multiple sources is a key requirement for many applications. Distributed data integration may be required for various reasons, including autonomy-related reasons requiring that certain data is owned and managed locally or performance/scalability-related reasons where multiple resources are used to enable parallel and distributed data processing, as exemplified in the context of relational data integration by OGSA-DQP [http://www.ogsadai.org.uk/dqp]. The WS-DAI-RDF(S) specifications should allow data integration applications to be built on top of WS-DAI-RDF(S) data resources by providing the necessary interfaces and access patterns for efficient distributed data integration. Data integration applications can benefit from the seamless integration with other capabilities inherent to service-based Grids such as resource discovery and resource monitoring, as demonstrated by OGSA-DQP and OGSA-DAI-RDF [OGSA-DAI-RDF].

Goal

Given a set of distributed RDF data sources, the goal is to provide a robust and scalable federated database that supports seamless access over the heterogeneous RDF database management systems.

Requirement Analysis

Distributed data integration applications require a common interface to eliminate syntactic heterogeneities that may exist between individual data resources. Syntactic heterogeneities may be present in both the interfaces used to retrieve information about the data held by a data resource and the interfaces used to submit queries and retrieve data from resources. Data retrieval mechanisms must support large datasets and allow a client to have control over the rate at which data is delivered to avoid being swamped with data from multiple sources. Third-party delivery, supported in an indirect fashion by the WS-DAI-RDF(S) specifications, is also important as in some cases where it is possible to perform data integration using multiple computational resources. Third-party delivery allows a data integration application to issue sub-queries to data resources and delegate the various tasks involved in processing the data to other services, allowing the data to be integrated in parallel.

Use Case

Ubiquitous code (ucode) [UCODE] identifiers, often physically implemented as RFID tags, can be used to identify real-world objects among multiple computer systems. A ucode relation (UCR) is a relationship between objects (identified by ucodes), which can be modeled as an RDF triple, for example, this apple (subject ucode) is produced by (predicate relation ucode) the JA Tsugaru-Minami Farm (object ucode). UCR triples are generally stored in a wide area distributed databases (UCR databases), and there have been efforts to implement UCR databases that support SPARQL endpoint, e.g. by Nihon Unisys [http://dev.tyzoh.jp/trac/semi-structured-db/].

Figure 8 provides an overview of a grid-based distributed RDF database, which federates various (UCR) RDF databases. The service-based SPARQL query interfaces provide a uniform access mechanism to the heterogeneous RDF databases for implementing distributed query processing over the individual data resources.

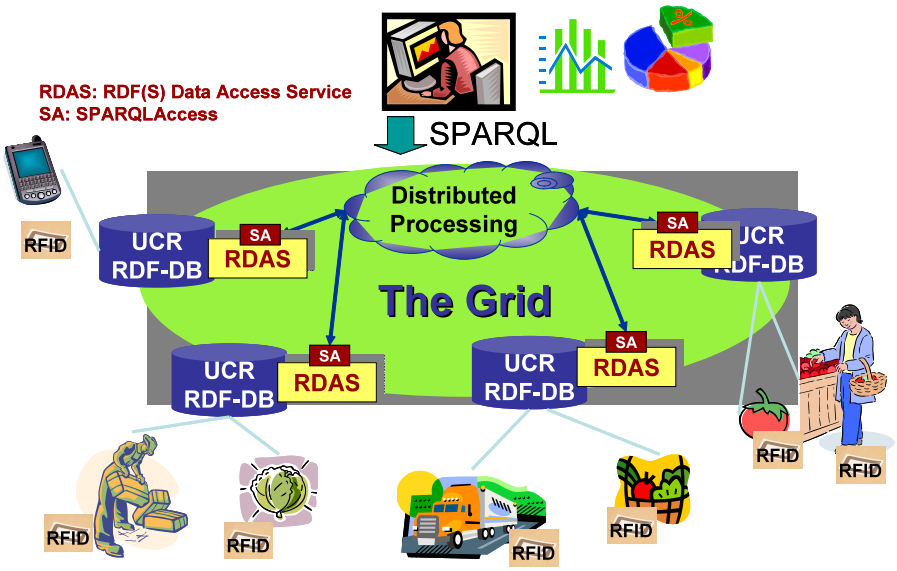

Figure 8: Large scale distributed RDF database

In this scenario, a federated SPARQL query is decomposed into a number of sub-queries, based on the information that has been retrieved about the properties of each resource. Sub-queries are sent to the individual data resources and the results are integrated in order to answer the federated query. The standardized interfaces provided by the WS-DAI-RDF(S) specifications mean that syntactic heterogeneities present amongst the individual data sources are resolved when performing these tasks. Data integration is performed by multiple computational resources within the area labeled “distributed processing” in the figure. The indirect data access pattern (SPARQLExecuteFactory operation) is used to execute each sub-query, which results in the creation of a new data resource for each set of query results. The various data integration tasks (e.g. joins, unions etc.) that need to be performed are then delegated to appropriate nodes in the set of computational resources, which are given references to the created data resources that need to be accessed in order to perform their allocated tasks. The SPARQLResultsSetAccess port-type’s GetResults operation allows results to be pulled from data resources as they are needed, enabling the computational nodes

implementing the distributed query processing to control the rate at which data is retrieved from data resources. The use-case therefore relies extensively on the indirect data access pattern supported by WS-DAI specifications.

4.4. ADMIRE Registry

Motivation

The EC funded ADMIRE[1] (Advanced Data Mining and Integration for Europe) project aims to build a platform that will bridge the gap between domain experts and the application of Data Mining and Integration (DMI) technologies to these domains. The goals and motivations for ADMIRE, as well as a more complete description of this use case, can be found in [ADMIRE]. In ADMIRE, registries are used to track processing elements (PE) – these are primitive or composite software components that encapsulate DMI algorithms. PEs are created by data mining experts and are then made available to the ADMIRE community. A registry allows users to locate these PEs when required using SPARQL. There are two main levels at which these registries operate: a local (or workbench) level in which a DMI Workbench tool is used to create PEs which are then stored in this local registry; there is also registries (accessed through a gateway) populated by the local registries with the PEs developed at other workbenches and made available to other ADMIRE communities.

Goal

In ADMIRE registries play a key role. A registry stores the location of the PEs and a description of them. This description contains the data types of the input and output parameters for each PE and any restrictions associated with these. Information about the PEs is encoded using RDF and stored in the ADMIRE registry. In ADMIRE, DMI expert users create these PEs, store them in a repository and update the registry, first locally and then this may be optionally migrated to registries. To retrieve these PEs users have to use the SPARQL query language receiving as result the location of those PEs. The goal of these registries is to provide users with a way of retrieving the PE information that meets their requirements to execute DMI workflows using the PEs, either created by them or other users, within the ADMIRE community. They are accessed by different users at different times in different contexts (binding the states to the users), thus a way for managing them is necessary and it is provided by the WS-DAI-RDF(S) specification.

Requirement Analysis

The registry allows the data stored in it to be queried, updated and added to. It is thus necessary to have methods available for querying and storing RDF data. New ADMIRE nodes may be incorporated into the system at certain points, therefore the local registry at each workbench will have to update (if allowed by the workbench users) the registries.

Use Case

Figure 9 represents the interactions of ADMIRE users querying the registries (both local and global) using SPARQL.

[1] Framework 7 ICT 215024.

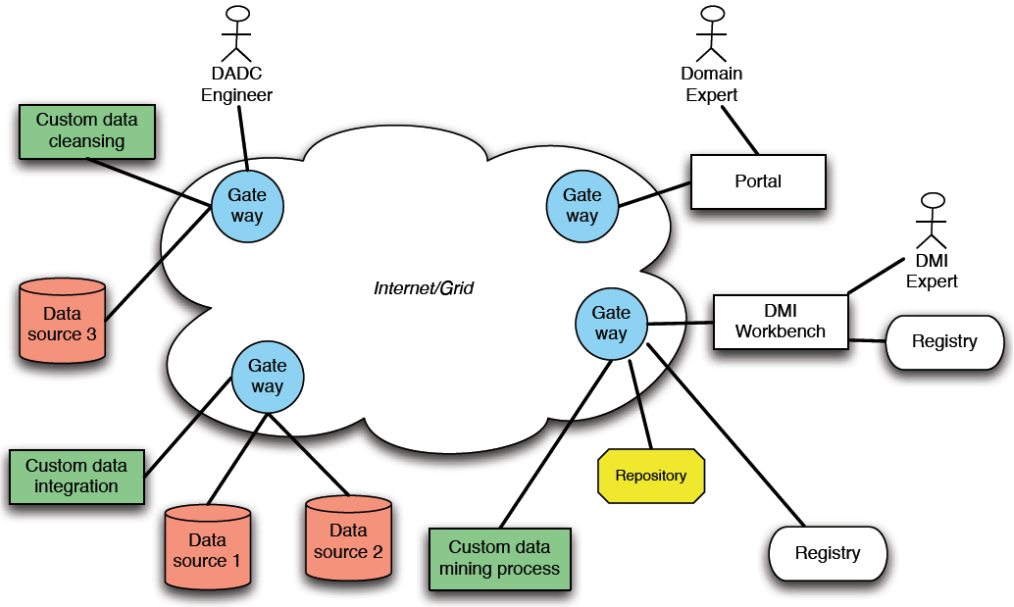

Figure 9: ADMIRE use case: Data-Aware Distributed Computing (DADC) engineers, DMI experts and Domain experts acting as end users for and ADMIRE community. For more details consult with [ADMIRE]

The ADMIRE registries provides access to the PEs that are designed by the ADMIRE end users. Both registries are accessible via interfaces based on the WS-DAI-RDF(S) specification. The registries are also accessible via an implementation based on OGSA-DAI [http://ogsadai.org.uk] activities. The latter is used by the ADMIRE workbench users whilst the former is used by users external to the project to query the registry for available PEs – hence the requirement to use a standards based mechanism.

The working process is described as follows:

- a user queries the local registry for a Processing Element (PE) by sending a SPARQL query;
- if there is a PE that matches the query in the local registry then this is returned to the user else the local registry queries the other ADMIRE registries for possible matches. If there is a match, the location of these matches are propagated to the local registry;
- the local registry returns the locations of the desired PEs to the user if any exist;
- the user accesses and executes the PE stored in the repository containing the data mining models.

It is important to note that there are two registries with the same functionality but working at different levels. The first (local) registry is located locally for a user to access directly. The other registries are available for external queries from the local registries (via the gateways) and are populated by the local registries. The content of the registries will be available to people external to ADMIRE using the standard WS-DAI-RDF interfaces.

4.5. Summary

From these four scenarios it is clear that there is a real benefit to having a standardized set of interfaces to be able to access RDF(S) data. Moreover, in contrast to the existing specifications for accessing RDF(S) data resources, such as the SPARQL protocol, the different types of access patterns provided by the WS-DAI family of specifications can provide additional ways of addressing the scenarios. For example, this is demonstrated in Section 4.2 by the Grid Resource Annotation and Monitoring use case, which requires the ontological access primitives

provided by the WS-DAI-RDF(S) Ontology specification, and in Section 4.3 by the Federated SPARQL use case which requires the indirect access pattern supported by WS-DAI specifications. Thus we believe that there is a real need for this type of functionality both within the grid world and more generally within the RDF(S) communities as well.

5. Conclusion

The provisioning of RDF(S) access mechanisms is of major interest to the OGF community as it will be the first step in the way for enhancing the current grid by means of semantic technologies.

The DAIS WG is engaged in an initiative for providing such mechanisms as part of the data access and integration facilities that are being defined at the moment. The work will be carried out in parallel: one focused on accessing following an ontological approach, and the other targeted at accessing to RDF(S) contents using the query language.

The work that is to be accomplished has its roots is previous work undertaken by AIST[2] and the OntoGrid[3] project, teams who will keep working in these issues and will lead the initiative.

We encourage the rest of the DAIS WG members, OGF members and Semantic Grid experts who are interested in the forthcoming work, to join the initiative.

Author Information

Isao Kojima
Information Technology Research Institute
National Institute of Advanced Industrial Science and Technology (AIST)
Central 2, 1-1-1 Umezono, Tsukuba, Ibaraki
305-8568
Japan
email: kojima@ni.aist.go.jp

Miguel Esteban Gutiérrez
Ontology Engineering Group (OEG),
Universidad Politécnica de Madrid (UPM),
Campus de Montegancedo s/n,
28660 – Boadilla del Monte, Madrid
Spain
email: mesteban@fi.ump.es

Oscar Corcho
Ontology Engineering Group (OEG),
Universidad Politécnica de Madrid (UPM),
Campus de Montegancedo s/n,
28660 – Boadilla del Monte, Madrid
Spain
email: ocorcho@fi.upm.es

Steven Lynden
Information Technology Research Institute
National Institute of Advanced Industrial Science and Technology (AIST)
Central 2, 1-1-1 Umezono, Tsukuba, Ibaraki
305-8568
Japan
email: steven.lynden@aist.go.jp

Mario Antonioletti

[2] http://www.aist.go.jp
[3] http://www.ontogrid.eu

EPCC,
JCMB,
The King's Buildings,
Mayfield Road,
Edinburgh EH9 3JZ,
United Kingdom.
email: mario@epcc.ed.ac.uk

Carlos Buil Aranda
Ontology Engineering Group (OEG),
Universidad Politécnica de Madrid (UPM),
Campus de Montegancedo s/n,
28660 – Boadilla del Monte, Madrid
Spain
email: cbuil@fi.upm.es

Said Mirza Pahlevi
National Institute of Advanced Industrial Science and Technology (AIST)
Current Affiliation:
Institute of Statistics(STIS)
Jl. Otista 64c, Jakarta, 13330
Indonesia
email: mirza@stis.ac.id

Asunción Gómez-Pérez
Ontology Engineering Group (OEG),
Universidad Politécnica de Madrid (UPM),
Campus de Montegancedo s/n,
28660 – Boadilla del Monte, Madrid
Spain
email: asun@fi.upm.es

Contributors

Masahiro Kimoto, Business Search Technology Corporation, Japan.
Norman W Paton, University of Manchester

Acknowledgements

We would like to thank to those members of the DAIS-WG who have helped us in the process of chartering the initiative as part of the DAIS WG: Malcolm Atkinson, Amy Krause, and Dave Pearson.

We also have to thank AIST and the OntoGrid project, as their funding has made possible this work.

OMII-UK resources contributed to the production of this work. OMII-UK is funded by EPSRC through the UK e-Science Core Programme and through the JISC.

Intellectual Property Statement

The OGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this recommendation. Please address the information to the GGF Executive Director.

Full Copyright Notice

References

[ADMIRE]

M. Atkinson, P. Brezany, O. Corcho, L. Han, J.I. van Hemert, L. Hluchy, A. Hume, I. Janciak, A. Krause, and D. Snelling. ADMIRE White Paper: Motivation, Strategy, Overview and Impact. Technical Report version 0.9, University of Edinburgh, January 2009

[DAIRDFS]

M. Esteban, I. Kojima, S. Mirza, O. Corcho and A. Gomez, *Accessing RDF(S) data resources in service-based Grid infrastructures.* Concurrency and Computation: Practice and Experience, Vol.21.No.8, 1029-1051 (2009)

[GLUE]

S. Andreozzi et al. GLUE Specification v.2.0, OGF, GLUE Working Group, March 2009 http://www.ogf.org/documents/GFD.147.pdf.

[GRID]

I. Foster and C. Kesselman (Eds.) *GRID 2, Blueprint for a New Computing Infrastructure*, Morgan-Kaufmann Press. (2003)

[GRID2SEM]

C. Goble, D. DeRoure, N. Shadbolt and A. Fernandes, *Enhancing services and applications with knowledge and semantics*, in [GRID] (2003)

[MATCH]

Said Mirza. *A Semantic Matchmaker for RDF/OWL-based Service Repositories*, RDF, Ontologies and Meta-Data Workshop, UK National e-Science Institute (2006)

http://www.nesc.ac.uk/action/esi/download.cfm?index=3183

[OGSA-DAI-RDF]

I. Kojima. *Design and Implementation of OGSA-DAI-RDF,* 3rd GGF Semantic Grid Workshop

http://www.semanticgrid.org/GGF/ggf16/slides/Design%20and%20Implementation%20of%20OGSA-DAI-RDF.ppt

[RDF-CONCEPTS]
G. Klyne and J. J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax, W3C Recommendation. 10 February 2004 http://www.w3.org/TR/rdf-concepts/

[RDF-SEMANTICS]

P. Hayes (Ed). *RDF Semantics*. W3C Recommendation 10 February 2004

http://www.w3.org/TR/rdf-mt/

[RDFS]

D. Brickley and R.V. Guha, *RDF Vocabulary Description Language 1.0: RDF Schema.* W3C Recommendation 10 February 2004

http://www.w3.org/TR/rdf-schema/

[RDF-XML]

D. Beckett (editor), RDF/XML Syntax Specification, W3C Recommendation, 2004.

http://www.w3.org/TR/rdf-syntax-grammar/

[RESULTS]

D. Beckett, *SPARQL Query Results XML Format*, W3C Recommendation 15 January 2008.

http://www.w3.org/TR/rdf-sparql-XMLres/

[S-MDS]

S. Mirza and I. Kojima. *Towards Automatic Service Discovery and Monitoring in WS-Resource Framework,* In: Proc. of the First International Conference on Semantics, Knowledge and Grid. (2005) 932-938

[S-OGSA]

P. Alper, S. Bechhofer, O. Corcho, C. Goble, I. Kotsiopoulos, P. Missier. *An overview of S-OGSA: a Reference Semantic Grid Architecture.*Journal of Web Semantics: Science, Services and Agents on the World Wide Web, vol. 4, no. 2, June, 2006, pp. 102-115 (DOI: 10.1016/j.websem.2006.03.001).

[SPARQL]

Eric Prud'hommeaux and Andy Seaborne. *SPARQL Query Language for RDF*, W3C Recommendation 15 January 2008

http://www.w3.org/TR/rdf-sparql-query/

[SPROT]

K. Clark, *SPARQL Protocol for RDF*, W3CRecommendation 15 January 2008.

http://www.w3.org/TR/rdf-sparql-protocol/

[S-SRB]

S.J. Jeffrey, and J. Hunter, *A Semantic Search Engine for the Storage Resource Broker*, in *3rd GGF Semantic Grid Workshop*. 2006: Athens, Greece.

[SW]

T. Berners-Lee, J. Hendler, O. Lassila. *The Semantic Web*. Scientific American, 284(5) (2001) 34 – 43

[SUPDATE]

A. Seaborne et al. SPARQL Update - A language for updating RDF graphs. W3C Member Submission, 15 July 2008

http://www.w3.org/Submission/2008/SUBM-SPARQL-Update-20080715/

[TANGDK]

H. Tangmunarunkit, S. Dekker and C. Kesselman. *Ontology-based resource matching in the grid- The grid meets the semantic web,* Second International Web Conference, ISWC2003 (2003)

[UCODE]

T-Engine Forum, "Ubiquitous ID Architecture", UID-CO00002-0.00.24, Nov. 2006 (in Japanese).

[WS-DAI]

M. Antonioletti, M. Atkinson, A. Krause, S. Malaika, S. Laws, N. W. Paton D. Pearson, and G. Riccardi. *Web Services Data Access and Integration – The Core (WS-DAI) Specification, Version 1.0*. GWD-R, Global Grid Forum, DAIS Working Group, Jun 2006.

[WS-DAIR]

M. Antonioletti, B. Collins, A. Krause, S. Laws, J. Magowan, S. Malaika, and N.W. Paton. *Web Services Data Access and Integration – The Relational Realisation (WS-DAIR) Specification, Version 1.0*. GWD-R, Global Grid Forum, DAIS Working Group, Jun 2006.

[WS-DAI-RDF(S)-Ont]

M. Esteban and A. Gomez: *Web Services Data Access and Integration - The RDF(S) Realization(WS-DAI-RDF(S)) RDF(S) Ontology Specification, Profile 0,*Open Grid Forum, DAIS Working Group

http://forge.gridforum.org/sf/go/doc15613?nav=1

[WS-DAI-RDF(S)-Query]

I. Kojima, S. Mirza and S. Lynden: Web Services Data Access and Integration – The RDF(S) Realization(WS-DAI RDF(S)) RDF(S) Querying Specification, , Open Grid Forum, DAIS Working Group

http://forge.gridforum.org/sf/go/doc14074?nav=1

[WS-DAIX]

M. Antonioletti, S. Hastings, A. Krause, S. Langella, S. Laws, S. Malaika, and N.W. Paton. *Web Services Data Access and Integration – The XML Realization (WS-DAIX) Specification,Version 1.0*. GWD-R,Global Grid Forum,DAIS Working Group, Jun 2006.

[WSDL]

E. Christensen, F. Curbera, G. Meredith and S. Weerewarana. Web Services Description Language (WSDL) 1.1, W3C Note. 15 March 2001

http://www.w3.org/TR/wsdl

[WS-ResourceLifetime]

L. Srinivasan and T.Banks. Web Services Resource Lifetime 1.2 (WS-ResourceLifetime). OASIS Standard, 1 April 2006.
http://docs.oasis-open.org/wsrf/wsrf-ws_resource_lifetime-1.2-spec-os.pdf

[WS-ResourceProperties]

S. Graham and J. Treadwell. Web Services Resource Properties 1.2 (WS-ResourceProperties). OASIS Standard, 1 April 2006.
http://docs.oasis-open.org/wsrf/wsrf-ws_resource_properties-1.2-spec-os.pdf

[WSRF]

T. Banks (editor), Web Services Resource Framework (WSRF) - Primer v1.2. Oasis Standards, May 2006

http://docs.oasis-open.org/wsrf/wsrf-primer-1.2-primer-cd-02.pdf