DODA 0.1: AN ONTOLOGY FOR LIGHTWEIGHT INTEGRATION OF SEMANTIC DATA ACCESS TECHNOLOGIES

Giovanni Tummarello[1], Christian Morbidoni[1], Michele Nucci[1], Richard Cyganiak[2]

[1] SEMEDIA - Dipartimento di Elettronica, Intelligenza Artificiale e Telecomunicazioni
Università Politecnica delle Marche, Ancona (ITALY)

[2] Institut für Produktion, Wirtschaftsinformatik und OR
Freie Universität Berlin (GERMANY)

Abstract

Over the years, a number of studies and practical efforts have proposed technologies to retrieve, publish, cooperatively edit, lookup and query RDF models. Also, other studies and practices have shown that standard communication channels, e.g. e-mail, RSS and IM, can be successfully used in Semantic Web scenarios. In this paper we collectively refer to such technologies as "Semantic Web data access" and present the "description of data access" ontology (DODA). DODA is a lightweight formalization which features a simple structure and specifically targets Social Semantic Desktop use cases. DODA supports client software in providing meaningful and assisted user interfaces; as a result, the software con provide the user with a seamless view of heterogeneous data sources and facilitate giving external access to local data.

1. Introduction

In this work we introduce the Description of a Data Access (DODA) formalization. DODA is a lightweight ontology to describe sources and channels for RDF and simple data access services related to Semantic Web scenarios. Descriptions expressed in DODA enable software clients to understand the basic functionalities of heterogeneous semantic channels and present them to the user in a uniform way, e.g. according to their capabilities and intended purpose. The idea is to provide a mean to express, for example, that a given URL contains an RDF dump, that this is updated at a given expected time interval, and is called "about Alice's projects". Similarly, one can express that a a given URL provides an RSS feed named "pictures" with semantic attachments, or that RDF models as attachments are processed if sent to a specific email address (i.e. email as personal incoming Semantic Web channel). Finally, DODA can be used to enhance WSDL/RDF descriptions with indications that enable a general web service to participate to the scenarios that DODA aims to facilitate.

In the first part of this paper we will give definitions and present the scenario and the design goal of the ontology. In the second part we provide a review of RDF communication channels either specifically meant to handle RDF (e.g. URIQA or RDFGrowth P2P algorithm) or that can be someway used for it, especially in Semantic Desktop scenarios (e.g. semantic email). In the third part we define the specific features of the DODA ontology. Finally, we provide examples of FOAF and DOAP files enhanced with DODA constructs.

2. Supporting a Social Semantic Desktop scenario

DODA is inspired by the requirements of the "Social Semantic Desktop" scenario. To explain such requirements and associated use case, lets consider tools such as the forthcoming Nepomuk [1], DBin [2] and Haystack [3]. In this paper we will refer to these applications *as Semantic Personal Knowledge Managers (SPKM)*. SPKMs are rich local applications which handle semantically structured knowledge either created by users, extracted from local resources, retrieved or exchanged with remote sources or peers.

SPKM applications usually provides:

- A Semantic repository

- Advanced (e.g. adaptive with respect to the context) GUI over the local RDF knowledge. Both visualization and editing are supported. Editing may include simple tagging as well as editing of complex RDF structures.
- Readers and writers for a variety of information channels.
- Host for a number of specific plug-ins

In DBin, for example, a user might import her local RDF data, edit it, make binary files such as photos available on-line automatically and describe them in the graph, share the resulting model and cooperatively edit it in RDF P2P groups, and create full or partial RDF dumps that can be posted manually or automatically as HTTP retrievable files or RSS feeds.
Semantic Desktop is a term chosen for the vision of a computing environment that offers practical and convenient ways for the user to employ semantic technologies in everyday information related activities. The idea is that semantics might help in better organizing and locating documents and other everyday information objects such as emails, contacts, and calendar events, regardless that they are physically stored locally or remotely on the Web. [4] gives an overview of the basic ideas behind the Semantic Desktop and its current state. SPKMs provide a great base for Distributed Semantic Desktop functionalities. Once such RDF representation of the local "desktop information space" is paired with appropriate semantic communication and data access channels across among users, the end result would be what has been called a *Social Semantic Desktop* [*5*].

2.1. The role of DODA: use cases and requirements

DODA aims at providing the minimum of formalism needed to set up an infrastructure that supports Social Semantic Desktop scenarios.
On-line resources described by DODA are likely to have a very simple logic, most of the time this simply being the fact that a RDF model is served, published, shared or shipped to one or more recipients, along with a human legible description which enables the user to decide whether that channel is of interest for the specific task at hand.
DODA therefore focuses on use cases in which a human is directly involved. It provides a level of formalization which enables social semantic desktop clients to present channels, heterogeneous in origin and kind, in a uniform way along with human readable information to support use cases such as these:

- Alice's personal FOAF[6] profile contains DODA information to specify the location of her semantic news feed that announces her new blog posts, photos, pictures and public calendar events. She also publishes an RDF representation of her shared documents folder, uploaded daily by her social semantic desktop software, but it is access restricted. Her profile also states that she has an incoming semantic channel: her semantic desktop client is configured to co-read her email, taking care of RDF attachments.

- Bob is in the audience at Alice's talk at a conference. He googles her and visits her homepage. It is equipped with FOAF auto-discovery[1]. Bob's social semantic desktop client is configured to automatically retrieve and display RDF information found on visited pages, similar to PiggyBank [7]. The FOAF file, including its DODA annotations, are processed. At this point, Bob can decide to subscribe to Alice's news feeds channel.

- Carl is a new co-worker of Alice. He opens her FOAF file in his client, configured to automatically process DODA channel descriptions: Alice is added as a potential source and destination of semantic data to his address book. Carl passes the Access Control List requirements to Alice's shared documents, so his client retrieves the dump and Carl sees it merged with his local knowledge. He sends a semantic email to Alice's incoming interface with his own profile, including the DODA indication of his channels. He also sends her the indication of an RDFGrowth channel, which is a P2P cooperative model, for their next project. Alice can now hook up to this channel.

1 http://rdfweb.org/topic/Autodiscovery

- Later, Alice visits the DBin project homepage. Her client auto-discovers the Description Of A Project (DOAP) file (footnote: our DOAP file). From there it discovers that the list of related publications is available as an RDF file, that project news are available as a semantic news feed, and that discussion about the project is available either in an RDFGrowth P2P group or as an archived RDF dump of the same. She forwards the DODA descriptions of the DBin news channels to the shared model so Carl will automatically know about it. She also publishes a blog post about it and attaches the description. Bob, who is subscribed to her blog, will thus learn about it as well.

- After the project has finished, Alice decides to enable public URIQA[8] access to the knowledge base accumulated during the project. The semantic desktop software will update her FOAF file with a DODA description of the new channel. Alice also sends the description to the input channels of those coworkers and colleagues in her address book that are able to receive semantic messages.

In most of these scenarios, existing technologies can be used for the actual data exchange. Applications and tools like PiggyBank and Semantic Bank[9], Annotea, SPARQL[10] endpoints, RDFGrowth[11], URIQA, RSS and others provides specific functionalities for exposing and consuming RDF payloads, being in fact semantic channels. The challenge is to discover where to find the data, or where to send a message, and using which protocol.

3. Established and emerging Semantic Web "Data Access Technologies": a factorization

Over the years, a number of studies and practical efforts have proposed technologies to retrieve, publish, cooperatively edit, lookup and query RDF models. Also, other studies and practice have shown that standard channels, usually meant for generic text messaging and untyped data attachments, can be used successfully in Semantic Web scenarios and especially within the context of Social Semantic Desktop. In this paper, we collectively refer to such technologies as Semantic Web "data access".
In the analysis performed in this section, we identifying distinctive tracts and features of the most representative such technologies. Such “decomposition and factorization” will form the bases for the construction of the DODA ontology.
Given the specific purpose of this ontology, we ignore differences in performance, scalability, complexity of implementation etc. We are instead interested in the peculiarities which distinguish these technologies from a high level usage point of view: signature of the high level API and semantics of the service.
To be noticed that the interface we extract here for each featured technology is not representative of the whole technology but just of the novel aspect over the other ones. We will see later, in detailing the content of the DODA ontology, how the actual instances of the specific technologies are instead modeled by giving each as many interfaces of different nature as needed to cover the complete functionalities.
The API and semantic elicitation will be expressed as follows:

SIGNIFICATIVE INTERFACE NAME: Description of the intended semantics
Input: Description of each input datatype and semantics
Output [on callback]: Description of each output datatype and semantics. Asynchronous interfaces might generate callbacks.

When not specified, Inputs and Outputs are Null, or merely control codes (e.g. An HTTP return code)

3.1. Plain Old Published Model (POPM)

The simplest access technology for an RDF model is the plain HTTP call to retrieve a serialized version of it, which might be an actual server side file or generated on the fly upon receiving the request. We call such sources "POPM" (Pronounced Pop-em), Plain Old Published Models. The usage API of a channel to a POPM, initialized using its URL, supports one operation:

POPM: an RDF model is available for web retrieval
Output*:* the RDF model

A number of more advanced technologies also offer this access modality. An RSS 1.0 feed, for example, can be seen as a simple published model. SPARQL endpoints can also provide complete models via specially crafted *construct* queries. Semantic Wiki engines such as Semantic MediaWiki [12] can also provide HTTP URLs which act as POPMs while generated from the parsed content of the page.

3.2. Semantic RSS

RSS feeds are lists of items, with associated metadata, usually listed in order of publication. This is the case both for its RDF-based flavors (e.g. RSS 1.0[13]) and the simple XML versions (e.g. RSS 2.0[14]). With either version it is possible to ship RDF payloads instead of or in addition to human-readable content. The main function of RSS is that of providing notifications about new content to users which have subscribed. While this has historically been achieved by the use of mailing lists, RSS seems to be better suited to the job. As it is based on client side polling of a publicly available feed, this method is perceived as less invasive than subscribing to a mailing list, which usually requires email confirmations and idiosyncratic ways of unsubscribing.

While RSS usually gives no guarantee of how long a published item will be available in the feed, sometimes these are comprehensive lists of items. This the case of projects such as OpenAcademia [15], an RDF powered Internet application which enables users to get personalized RSS with lists of publications. Such RSS feeds which can be used as usual, but are especially suited in social semantic desktop purposes (e.g. exchanged between colleagues) or simply to drive the list of publications on one's homepage. The ability to create personalized feeds is given via a user accessible interface. Under a machine accessible point of views, OpenAcademia feeds are regular RSS 1.0 so OpenAcademia does not constitute a separate case. Similarly, and for the purpose of our analysis the important part is the the ability of notifying the receiver, rather than the information that the feed contains, which can be considered as a POPM.

Notification Interface: A notification from a usually public source of information of a new item

Output on Callback: an RDF representation with a guaranteed list of items

3.3. Atom Publishing Protocol, Semantic Bank, DBin Data Publishing Service

On top of what RSS provides, the Atom Publishing Protocol provides a way to publish new contributions or edit existing ones. With respect to our analysis and scope, the high-level API (Atom PostURI, EditURI functionalities) is identical to the interface provided by DBin's "Data Publishing Service"(footnote), or could be straightforward implemented with other technologies such as WebDAV, FTP etc.

Simile's Semantic Bank[9] has an API for storing a model into a repository given a URI of the same. The PiggyBank[7] client retrieves RDF by screen-scrapings of web pages or via embedded POPMs, then on user request uploads the model into Semantic Bank, which retains the name of the publisher. If the authentication part is hidden then Piggy Bank offers a simple publishing service for models. Basically same as Atom publishing would do, but without assigning URI.

From this discussion we conclude that binding details apart, over POPM and RSS these technologies suggest the following capabilities:

Publish/Replace: the model is made available to the public, retrievable as POPM

Input: Model , URL|void

Output: URI | void

3.4. Annotea / URIQA

Annotea [16] is a web based system which enables users to attach metadata to a Web page or to a part of it. The metadata is stored in specialized servers which take care of retrieving it once the user browses a previously annotated resource. Regardless of the specific use case that it was meant to address, Annotea is generic in scope as it enables a client to submit a generic annotation, as long with a specific indication of which resource which is to be considered the "main topic" of it. Similarly, Annotea can be queried with a URI to obtain a set of RDF models which are "about" that resource. Decomposing Annotea's functionalities, one could say that at publishing level it can be though equivalent to a *Publish/Replace* interface, if one

consider acceptable that a model is reposted somewhere, or to something as follows;

Annotea Publish: Informs a lookup service that a POPM is available "about" a specified URI
Input: URI , URL
Output: URL[]

Annotea Lookup: returns a list of POPM which are considered related to the resource
Input: URI
Output: URL[]

URIQA is similar as far as interface is concerned, but the semantic differs. URIQA presupposes the existence of a base model and enables the client to "peek" into it asking "about" a resource. To return a result, the URIQA server extracts from the well specified database a set of triples which it considers related to the request, usually forming a Concise Bound Description. At interface level, we can consider URIQA a sub-case of Annotea, one that returns a single authoritative graph instead of an arbitrary number of non-authoritative graphs.

URIQA lookup: The knowledge about a URI in a specific model is returned.
Input: URI
Output: RDF model

3.5. RDFGrowth / DBin

RDFGrowth is the P2P technology currently featured in the DBin project[2]. Using RDFGrowth[11], groups of users can synchronize RDF models about topics of common interest. By using an underlying monotonic model plus a revision system based on digital signatures, it is possible for group participants to keep different local "views" on what should go in the model while still contributing to and taking contribution from others. With respect to the scope of this work, an RDFGrowth group can be seen as having an interface which is a POPM (emulated by joining a group and collecting the knowledge from the others), as well as a public publishing interface. The Semantics is however different and requires a standalone category, as it is allowed for a peer to "revoke" statements said by others etc.

Share: a model is shared, it might be edited by others, local modifications are reflected remotely
***Input*:** RDF model (reference)
***Output on Callback*:** RDF model (reference), URIs which involved changes or RDF patches

3.6. Semantic Email / Semantic IM

Email and Instant Messaging (IM) convey text and possibly binary attachment from in a person to person basis. E-mails, in particular, have been investigated as a channel to exchange semantic recommendations in [17], where a Thunderbird plug-in is described which automatically attach meaningful RDF data to e-mails. Characteristic peculiarity of these communication means is that they imply direct involvement of the receiver, which means that the sender also knows he will have some attention by the human receiver. Once an email (or IM) address is known (e.g. one has learned that a receiver is capable of processing semantic attachments), the interaction interface can be simply modeled as the ability to send a model and the implicit sending of one's own email or IM address, which can be expressed as a URI.

An outgoing predefined email: A model is sent and brought to the attention of the other side
Input: RDF model, own URI Output: none

One's own email: A channel for incoming models sent by senders
Output on Callback: an RDF model, a URI (sender)

3.7. SPARQL endpoint

As a result of the work of the Data Access W3C Working Group (DAWG), the SPARQL protocol and query

language[10] is now in an advanced standardization phase. While limitations exist, SPARQL is a powerful query language, and it draws from previous experiences in the database area and of early adopters of semantic technologies.
SPARQL can provide ways of accessing a remote model which range from transferring the whole remote model (a sub-case of the CONTRUCT kind of query) or one of the many "named" models a SPARQL enabled knowledge base might contain. SPARQL CONSTRUCT queries can also transfer pieces of models either as they are in the original model or transformed.
SELECT queries will produce a table of variable bindings as output. Although such variables have a name, they do not have a predefined or machine interpretable semantics that could be interpreted without looking at the originating query. SPARQL also supports DESCRIBE queries which have similar semantics to the URIQA metadata-get request.

SPARQL construct: a graph extracted or transformed from the knowledge of a remote DB
Input: CONSTRUCT query
Output: RDF model

SPARQL select: executes the query remotely. The semantics of the result varies according to the query.
Input: SELECT query
Output: structured information (SparQL XML)

3.8. Edutella and Publish/Subscribe

The Edutella P2P distributed querying system[18] is dedicated to providing access to a network of database hosts which are willing to answer semantically structured queries. Successive modifications have shown how it is possible to have service servers enable the use of "long standing queries" which express the interest for a client peer to know when a new item with the given characteristics is available at some of the database peer, without requiring continuous "refresh" queries. If the requesting client peer is not logged in at the time a new result appears, this is stored and later notified when the client reconnects.
For the purpose of this analysis, direct queries have the same general interface of a SPARQL endpoint, albeit requiring a different query language.
The new element is the ability to provide a query as a parameter to later receive notifications, similar to RSS feeds produced directly by a query.

Edutella Query Publish Subscribe: long standing query results notification
Input: a query
Output on Callback: a set of URI of matching elements and URI of sources

4. The DODA ontology

The formalization in input/output APIs as summarized in the previous chapter forms the bases of the Description Of Data Access ontology. The center idea is to model a set of "data access technologies", e.g. the POPM concept, and enable one to declare the existence of a "deployment" of one such technology.
Since the tools for description are on purpose coarse grained, modeling a technology is not difficult. This task is furthermore seldom necessary as DODA includes instances which model many well known data access technologies (including the ones considered in the previous chapter). Using DODA to express that a deployment of a specific technology is made available at a specified address is simple: one just needs to create an instance of the *DODAInstallation* class connected with the preexisting instance of the appropriate technology within DODA. In general the URI of the installation instance will be a URL where the service is being offered. In such a simple case a single triple is all it takes:

```
<http://myBlog/myFeed>  doda:isDeploymentOf  doda:RSSNewsFeedTechnology .
```

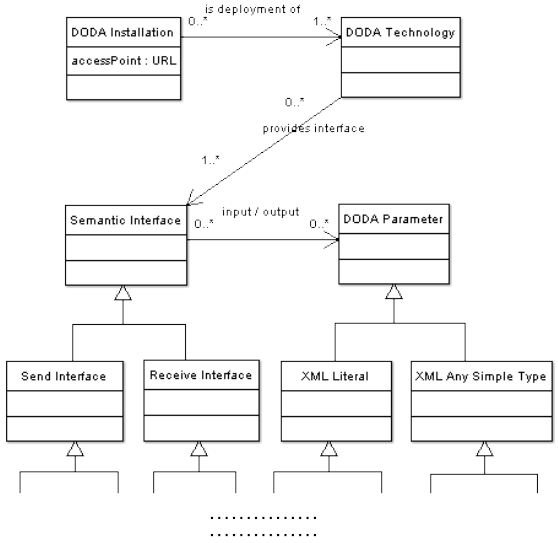

Figure 1: Main classes and relationships in DODA

The UML diagram in Figure 1 shows the main classes in DODA and their relationships. A *DODATechnology* (e.g. a POPM) can expose one or more interfaces, expressed as instances of the *SemanticInterfac*e class hierarchy. Figure 2shows the *SemanticInterface* hierarchy together with premodelled instances (smaller dots). Such instances are connected by *providesInterface* properties, to the appropriate instances of *DODATechnologies* (e.g. The *doda:RSSNewsFeedTechnology)* .

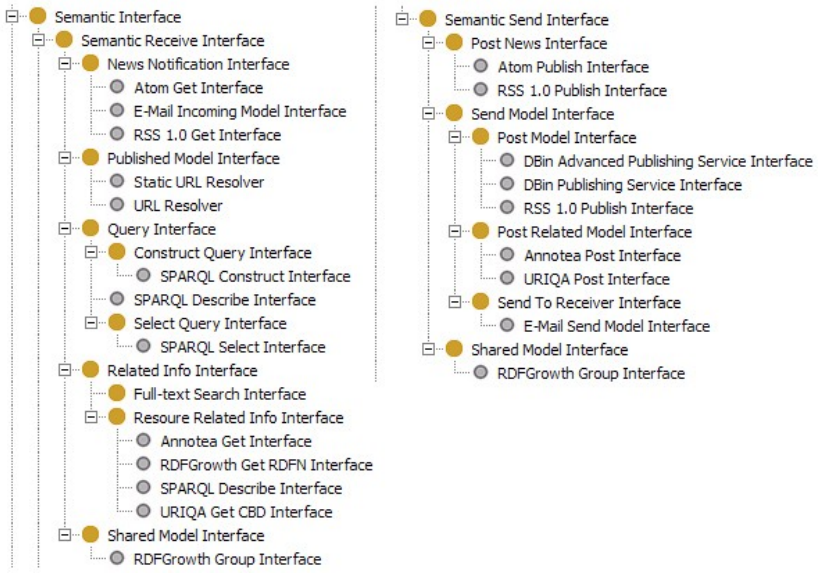

Figure 2: The SemanticInterface hierarchy and built in interfaces instances.

The class of the interface (e.g. for a POPM, is a *PublishedModelInterface* class) provides the semantic interpretation. A specific class might say, for example, that its instances are meant to retrieve RDF models by name, or , as a second example, to search for resources 'related' to a given one.

The first semantic distinction the interface hierarchy makes is between "sender" and "receiver" interfaces. While both “sender” and “receiver” instances have inputs and outputs, the semantic interpretation of “sender” interfaces is that of transmitting or communicating information, while “receiver” interfaces are meant to serve as sources of information. The hierarchy then goes on with more specialized classes.

The API provided by each interface has input and output parameters with precise meanings. An *URIQAGetInterface* instance, for example, takes a URI that represents a request for information and returns a RDF model containing metadata related to such a URI; a *PublishedModelInterface* instance gets the last version of a model posted on-line. In DODA, as shown in [example 1], such 'semantic specifications' of the

APIs are modeled by means of the *input* and *output* properties hierarchies.
It is to be noticed that the specific technology names (e.g. URIQA in *URIQAGetInterface*) used to define names of interfaces and properties *do not* imply that that property or interface is used only to model that specific technology instance. Rather, the technology name serves as a placeholder for the specific characteristic behavior/role of that interface as elicited in the 'semantic technologies factorization process' made in the previous chapter.

Example 1:

```
:requestedURI rdfs:subPropertyOf :output ;
        rdfs:domain :RelatedInfoInterface ;
        rdfs:range :URIParameter .
:relatedModel rdfs:subPropertyOf :input ;
        rdfs:domain :ResourceRelatedInfoInterface;
        rdfs:range :RDFModelParamenter .
:RDFModelDefaultP a :URIParameter .
:URIDefaultP a :RDFModelParamenter .
:URIQAGetInterface a :ResourceRelatedInfoInterface ;
        rdf:label "gets the CBD of the requested URI"@en ;
        :relatedModel :RDFModelDefaultP ;
        :requestedURI :URIDefaultP> .
```

In the example 1 the *relatedModel* and the *requestedURI* properties are sub-properties of the more generic *input* and *output* properties. They are used by the *URIQAGetInterface* to specify the meaning of its input and output: the interface returns a model which the service states to be 'related' to the input URI. These predicates connect the interface instance with two parameters (*RDFModelParameter and URIParameter*). The datatype of parameters is specified by the class type; The hierarchy of typed parameters, shown in Figure 3, might not be the only possible alternative but the current design supports the use cases currently considered by DODA (e.g. An RSS 1.0 can be imported and processed as an RDF/XML but also as a simple XML)

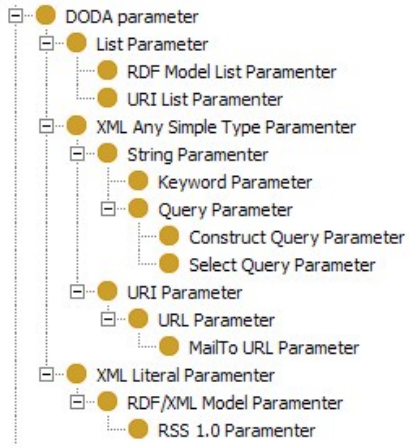

Figure 3: The DODAParamenter class hierarchy.

In the example, these parameter instances are built-in default values, however the service advertiser is free to create his/her own instance of *DODAParamenter* and to assign a meaningful URL to it.
This is done, for example, to map DODA input and outputs with web services descriptions written in WSDL WSDL/RDF[19] as illustrated in the following chapter.
In DODA, the user level semantics, expressed in the interfaces class hierarchy, is separated from the binding to a particular technology, specified by the interface instance, and as much as possible from the API (just the mandatory parameters are modeled as required properties, other might be added by specific instances of the interface). Decoupling these aspects is important for specifying clearly how the ontology is to be used and, at the same time, for limiting the complexity of the ontology, e.g. the depth of the class hierarchy.
Whenever a technology has aspects which are not captured by the means of the existing interfaces (e.g. both a way to send data and a way to receive), the modeling is done by creating more interfaces for it.

The ontology is currently available on the web[1], and it is expressed in OWL Lite. DODA use cases inside an implementing application (such as asking, e.g., which are the known installations of technologies which support the wanted interface and their parameter names) can be performed with a query language operating on top of a simple RDFS repository, that is without the need of a full featured reasoners. This fits the scope and lightweight nature of DODA, and enables its use in simpler environments such as hosted PHP applications.

4.1. DODA in use

To see DODA in use, there is clearly the need of an application supporting it. For an application to support DODA, it means to have single known *interface instance*s map directly to drivers that know how to use the specific technology. Drivers need not to be complex implementations, they can be as simple as a regular expression constructing an appropriate HTTP request.
Interface instances/drivers, on the other hand, should be reused when creating new instance of technologies. To model a read only Annotea services, for example, one creates a new technology but reuses the existing "receiver" interface. The *StaticURLResolver*, which represents simple HTTP get to obtain a model, is in fact reused in many instances of technology.
Coming back to the use cases sketched in section 2.1, we show here how the involved communication means can be described in DODA, thus be understood and made available to users by the client application.
In order to advertise her services and communication channel, Alice would need only to build upon predefined DODA instances which wrap existing technologies: *AtomCollection* for her news feed, *SimplePublishedModel* for the dump of her documents folder, *Email* for specifying that her mail folder supports Semantic Web data. Here is the N3 notation with the definitions of the deployments of such technologies. The URLs of the installations define the access point to the interfaces.

```
<mailto:alice@gmail.com> rdf:type doda:DODAInstallation ;
        doda:IsDeploymentInstanceof doda:Email ;
        rdfs:comment "You can send me RDF via e-mail here" .
<http://semedia.deit.univpm.it/dumps/29788973> rdf:type doda:DODAInstallation ;
        doda:IsDeploymentInstanceof doda:SimplePublishedModel ;
        rdfs:comment "Here you can find the RDF dump of my Documents folder" .
<http://semedia.deit.univpm.it/feeds/alice> rdf:type doda:DODAInstallation ;
        doda:IsDeploymentInstanceof doda:AtomCollection ;
        rdfs:comment "My public news feeds" .
```

In this examples we use some of the predefined technologies included in DODA. Lets consider the AtomCollection instance and its provided interfaces definition, built-in in DODA:

```
:DODATechnology a :AtomCollection ;
       :providesInterface :AtomPublishInterface;
       :providesInterface :AtomGetInterface .
:AtomPublishInterface a :PostNewsInterface ;
       rdfs:comment "Interface to post a the link to an RDF model using Atom."@en ;
       :hasInput :PublishedRDFModelURL .
:AtomGetInterface a :NewsNotificationInterface ;
       rdfs:comment "Interface for receiving notification about freshly published Atom news"@en ;
       :hasOutput :NewsListP .
```

When Bob imports Alice's DODA installations definition, his SPKM client will find out that the service available at *http://semedia.deit.univpm.it/feeds/alice* is a AtomCollection and exposes two interfaces, one of them is a *NewsNotificationInterface*. As the semantics of this interface is defined, the client can notify Bob that a new feed is available and show him the proper GUI for subscribing to a feed channel, but also have to check for the appropriate driver: this can be uniquely identified by the deployed instance (i.e. *AtomGetInterface*).
In the same way Alice's client can use the *AtomPublishInterface* definition to provide advanced publishing functionalities. Say that Alice is using a deployment of *DBinPublishingService* to publish its semantic blog postings. Since the interface provided by this technology has an output which exactly maps with the input of an AtomPublishInterface, Alice might automatically be suggested, after having posted a semantic blog, to advertise it on her personal feed at *http://semedia.deit.univpm.it/feeds/alice.*

1 http://semedia.deit.univpm.it/ontologies/doda.owl

A strong point of this approach is that of using the semantic interfaces abstraction to connect specific technologies and semantically defined, user level operations. The *GetRelatedInfoInterface*, for example, represents a way to search information related to a URI. Suppose Bob finds on-line a paper his interested in, then he might be able to ask his client application for a list of known services which can provide him with related metadata. The application, analyzing the DODA description which it owns, might present to Bob a URIQA services and/or RDFGrowth p2p group (as both have interfaces which are instances of *GetRelatedInfoInterface)*.
For a further example of practical DODA deployment see the advertisement of the news and discussion channel about the DBin project, as contained in the DBin Description of a Project[20] file:

```
.............
<http://dbin.org/news> rdf:type doda:DODAInstallation ;
        doda:IsDeploymentInstanceof doda:AtomCollection ;
        rdfs:comment "News about the DBin project" .
<http://semedia.deit.unimp.it/RDFGrowth/groups/DBinProject> rdf:type doda:DODAInstallation ;
        doda:IsDeploymentInstanceof doda:RDFGrowth ;
        rdfs:comment "An RDFGrowth group about the DBin project" .
<http://semedia.deit.unimp.it/RDFGrowth/groups/DBinProject/dump> rdf:type doda:DODAInstallation ;
        doda:IsDeploymentInstanceof doda:SimplePublishedModel ;
        rdfs:comment "An RDF dump containing a moderated version of what has been posted in the p2p
group within the last month" .
<http://public.dbin.org> rdf:type doda:DODAInstallation ;
        doda:IsDeploymentInstanceof doda:DBinPublishingService ;
 rdfs:comment "A service for publishing RDF graphs or retrieving them once the URL is known"
.................
```

4.2. DODA and Access Control

Most of the data access technologies modeled in the current version of DODA consider access control as an orthogonal facet - e.g. SPARQL protocol specifications say that implementations may impose several kind of limitations as needed but does not provide any detail. In general, most of these services are made available over HTTP so the related security mechanism would apply directly. In certain cases it might be useful to enrich a DODA descriptions to express such policies in RDF.
The ACL Schema[21] can be used to model access control to each interface of a specific installation of a technology by groups and single users.
Going back to our example use case, to express that just Carl can access Alice RDF dump of her "c:\work_documents\" folder, Alice could attach access control restrictions to her advertised installation node, meaning that the access to each interface provided by the deployed technology is restricted by the rule. This in practice would require that such rules are understood and enforced at least by Alice HTTP server, details of which are outside the scope of this discussion.

```
:SemanticInterfaceAccessRule a acl:ResourceAccessRule ;
        rdfs:label "Semantic Interface Access Privilege" ;
        rdfs:comment "Represents the privilege to write/read into/from a Semantic Interface" ;
        acl:hasAccessTo <http://semedia.deit.univpm.it/dumps/29788973>
        acl:access "Send News" ;
        acl:acessor :Giovanni, :Christian, :Michele .
:Christian a foaf:Person ;
        a acl:Identity ;
        foaf:name "Christian Morbidoni" ;
        foaf:mbox_sha1sum "c1d4cb076b0eac7d6dece499de92133b0af138f4"
        ex:publickey
<http://public.dbin.org/useraccounts/de46498/9523639c53e80192d8cfe09f11c36840.asc>.
```

In more advanced cases, one might want to provide a service, e.g. a SPARQL endpoint, granting public access to some of the interfaces (e.g. a simple SPARQL Describe query), while defining precise policies for others (e.g. a generic query interface, which can be more expansive to answer). Attaching access policies at single interface granularity is also supported by DODA with a specific access rule class.

4.3. DODA wrapping of RDF/WSDL descriptions

The Web Service Description Language is an XML based specification to describe interfaces of Web Services. Although sometime criticized for its complexity, WSDL has enjoyed increased support by tool vendors e.g. To provide automatic creation of code which can access WSDL described services. Mappings

from WSDL to RDF has been proposed as a way to make WSDL descriptions processable with semantic web technologies. DODA fits exactly this scenario, as it semantically enrich the bare Interface Definition provided by the base WSDL/RDF description. With a small number of RDF triples, it is possible to map WSDL interfaces to instances of DODA interfaces, and specific input output parameters with WSDL input and output messages. Interestingly, as WSDL specifies details about the binding. Applications need only to support a single DODA/WSDL driver to be able to automatically connect to any kind of DODA/WSDL described service. Details about such mappings are left to the DODA documentation available online.

5. Related Works and ontologies

Other than considering data access technologies, this work draws from and takes into consideration a number of other related formalizations, first of all the WSDL and OWL-D. While some part of DODA might resemble WSDL, DODA target and scope is well defined and supported by specific hierarchies, and premodelled instances. As we have seen in the previous section, rather than substituting WSDL, DODA integrates with it enabling WSDL/RDF described Semantic Web Data Access technologies to, e.g., participate in the use cases we have considered.
Under a semantic point of view, DODA presents a terse and specialized vocabulary of classes and specific instances within the specific task domain, as opposed to OWL-S which is general in scope but much more complex and therefore challenging with respect to use, deployment, interpretation and ultimately acceptance. The process happening inside DODA interfaces are simple to explain and our use cases of interest do not require that the machine understands the difference between said processes more than how made possible by DODA interface hierarchies. If needed however, it would be possible to associate OWL-S descriptions to DODA interfaces to make such inner models machine interpretable.
A number of other works are relevant and have been considered during DODA design. In [24] and [25] the importance of an "unified messaging" ontology is advocated. Messaging refer to Human to Human communications and the work focus is studying the way toward a unified view, for the end user, of "received and sent messages" no matter by which medium they were sent or received.
A similar concept, albeit with a more detailed and task specific ontology is presented in [26] where instant messaging is semantically enhanced with a descriptive ontology to model concepts such as "conversation" and the stream of individual messages. Such enhanced messages can also carry metadata, although this is still expected to be strictly related to the content of the message itself (e.g. a tag related to the message) rather than a generic channel for annotation exchange. In [27] the relationship between semantic web and blogging is explored and a system which enables the user to publish semantically enhanced RSS feeds is presented.
In [sharing context], the use of e-mails to exchange contextual information, along with documents, in a working group is discussed. The approach makes use of an extension of the FOAF vocabulary to semantically describe a group of users which share a context (a set of ontologies). A Firefox plug-in is described which enables users to automatically send, along with a document, an RDF file containing metadata related to the document (e.g. author, quotations, etc...) and to the specific context shared in the group.
Other works such as [28] have further shown interaction directly with desktop applications such as wikies[29]. Similar Use cases which are covered in DODA have been informally described in recent postings about "Subscribing to one's Brain" or "Life as RSS"[1], where it is reported to be of high interest to have a convenient way to manually select among aspects of semantically structured information produced by someone of whom one might have a high consideration.
Semantic Desktop and RSS has been explored in WonderDesk/WonderServer, part of the WonderSpace project, an e-Science tool. WonderDesk[30] uses RSS 1.0 to describe resources metadata and Hybrid solution (P2P[Jxta]/Server) as communication channel (WonderServer acts as a super-node of the network). WonderServer also acts as information aggregator for all WonderDesk peer in a specific group. In WonderDesk the RSS 1.0 vocabulary has been extended to describe specific metadata of various kinds of resource.

1 http://www.breakawayrepublic.com/blog/?p=40, http://www.ldodds.com/blog/archives/000217.html, http://ejohn.org/blog/life-as-rss/

5.1. Related Ontologies

Several ontologies exist which cover aspects somehow related to DODA or that can be used in conjunction with it. The Friend of a Friend (FOAF) as well as the Description of a Project (DOAP) vocabulary are commonly used to talk about persons and projects respectively, they can both make ready use of DODA extensions, e.g. to indicate data sources and contact channels.

EMiR[2], DOAML[3] and SIOC[31] are vocabularies for email, mailing lists, forums posts and usersmodeled respectively. WikiOnt[32] aims at integrating Wikipedia (and by extension other MediaWiki-based sites) into the Semantic Web, it describes the internal working of a wiki as a medium itself.

PIMO[33] introduces an ontology language that can be used to express personal mental models. It includes an simplified "domain upper ontology", but does not cover the specifically modeling of communication channels. Also potentially of interest, for future works, is the OWL Atom ontology[34], which models services provided trough this protocol.

6. Conclusion and Outlook

In this paper we give the following contributions. First we provide an ordered overview and an API/Semantic factoring of a number of technologies that have appeared in Semantic Web literature and practice and that might be considered either as "transport layer" or "data access services". Such factoring serves then as basis for the proposed DODA, ontology. DODA is an ontology for describing communication channels which groups them according to their peculiarity at the user interaction level. DODA enables a “lightweight” integration: it greatly facilitates the process and provides abstraction but, for example, expects humans in the end to decide, for example, which source among those which a client has learned about is meaningful to import or which export channel should be use to advertise a specific bit of information.

In our use cases, content publishers and communities use DODA to describe way by which they share, acquire, publish or give access to semantically structured information. As DODA comes loaded with a great number of pre-modeled well known technologies, using DODA can be as simple as adding a single triple to a FOAF or DOAP file.

Applications, on the other end, import such RDF descriptions and, supported by the ontology, can present proper user interfaces to interact with or interconnect channels. Such applications might be Semantic PKI as previously illustrated, but it can be foreseen DODA used in other situations, e.g., in server side software. While one might think such use cases to be far off futuristic, we believe they are actually quite simple once the proper technological infrastructure and UI is in place and provides assistance to the end user.

One of the most promising and awaited results of the Semantic Web initiative is the improvement of the way everyday information and knowledge is stored, retrieved and processed. To foster such such scenario, the ability of connecting to heterogeneous semantic data access services and channels published freely by individuals or organizations seems particularly important.

Simplicity and task specialization are the most important and interesting aspect in DODA stands out especially when confronted with complex proposal such as OWL-S; much like as happened with FOAF, which with its simplicity and task oriented nature, has undoubtedly received a lot of acceptance and ultimately widespread as opposed to more powerful yet complex vocabularies.

DODA 0.1 is currently made available on the web and further works will give the highest priority to gathering community feedbacks and practical validation.

References

[1] NEPOMUK - The Social Semantic Desktop, http://nepomuk.semanticdesktop.org/

[2] DBin project, http://dbin.org

[3] Haystack Project, http://haystack.lcs.mit.edu/

2 http://www.schemaweb.info/schema/SchemaDetails.aspx?id=46

3 http://www.schemaweb.info/schema/SchemaDetails.aspx?id=217

[4] Leo Sauermann, Ansgar Bernardi, Andreas Dengel, “Overview and Outlook on the Semantic Desktop”, 1st Workshop on The Semantic Desktop at the ISWC 2005 Conference, 2005
[5] Stefan Decker, Martin Frank, “The Social Semantic Desktop”, DERI Technical Report, 2004
[6] FOAF Project, http://www.foaf-project.org/
[7] Piggy Bank, http://simile.mit.edu/piggy-bank/
[8] The URI Query Agent Model, http://sw.nokia.com/uriqa/URIQA.html
[9] Semantic Bank, http://simile.mit.edu/semantic-bank/
[10] SPARQL Query Language for RDF, W3C Candidate Recommendation 6 April 2006, http://www.w3.org/TR/rdf-sparql-query/
[11] G. Tummarello, C. Morbidoni, J. Petersson, F. Piazza, P. Puliti, "RDFGrowth, a P2P annotation exchange algorithm for scalable Semantic Web applications", Mobiquitous, Boston, 2004
[12] Semantic MediaWiki, http://wiki.ontoworld.org/index.php/Semantic_MediaWiki
[13] RDF Site Summary (RSS) 1.0, http://web.resource.org/rss/1.0/
[14] RSS 2.0 Specification, http://blogs.law.harvard.edu/tech/rss
[15] Open Academia, http://www.openacademia.org/
[16] Annotea Project, http://www.w3.org/2001/Annotea/
[17] Stefania Ghita, Wolfgang Nejdl, Raluca Paiu , "Semantically Rich Recommendations in Social Networks for Sharing, Exchanging and Ranking Semantic Context" 2005 4th International Semantic Web Conference, Galaway, Ireland, 2005
[18] Wolfgang Nejdl, Boris Wolf, “EDUTELLA: A P2P Networking Infrastructure Based on RDF”, 11th International World Wide Web Conference, Honolulu, 2002
[19] Web Services Description Language (WSDL) Version 2.0: RDF Mapping, http://www.w3.org/2002/02/21-WSDL-RDF-mapping/, W3C Working Draft, 2006
[20] DOAP: Description of a Project, http://usefulinc.com/doap
[21] W3C ACL Schema, http://www.w3.org/2001/04/ACLS/Schema
[22] WITW: NSDL, http://norman.walsh.name/2005/03/12/nsdl
[23] SMEX-D, http://www.tbray.org/ongoing/When/200x/2005/05/03/SMEX-D
[24] The Message without the Medium: Unifying Modern Messaging Paradigms through the Semantic Web, http://citeseer.ist.psu.edu/bakshi04message.html
[25] Dennis Quan, Karun Bakshi, David Karger, A Unified Abstraction for Messaging on the Semantic Web, Proceedings of WWW, 2003
[26] T. Franz, S. Staab, SAM: Semantics Aware Instant Messaging for the Networked Semantic Desktop, Workshop on the Semantic Desktop at the ISWC, 2005
[27] D. R. Karger and D. Quan, “What would it mean to blog on the semantic web?”, Third International Semantic Web Conference, 2004
[28] K. M¨oller, U. Boj¯ars, and J. G. Breslin, “Using Semantics to Enhance the Blogging Experience”, Third European Semantic Web Conference, Montenegro, 2006
[29] R. Tazzoli, P. Castagna, S. Emilio Campanini, “Towards a Semantic Wiki Wiki Web”, Demo Session at ISWC, 2004
[30] Xiang Zhang, Wennan Shen, Yuzhong Qu, "WonderDesk - A Semantic Desktop for Resource Shating and Management", Workshop on The Semantic Desktop, ISWC 2005, Ireland, 2005
[31] SIOC Project, http://sioc-project.org/
[32] WikiOnt Vocabulary Specification, http://sw.deri.org/2005/04/wikipedia/wikiont.html
[33] PIMO - a PIM Ontology for the Semantic Desktop, http://www.dfki.uni-kl.de/~sauermann/2006/01-pimo-report/pimOntologyLanguageReport.html
[34] AtomOwl Vocabulary Specification, http://bblfish.net/work/atom-owl/2006-06-06/AtomOwl.html