Using the Rhizomer Platform for Semantic Decision Support Systems Development

Roberto García, Universitat of Lleida, Spain

Abstract. Decision support systems get more useful as they manage to make decisions more informed. However, the cost of information and of combining and making it available in the appropriate context make this a tricky trade-off. Fortunately, Semantic Web technologies make it possible to easily publish and reuse data. But this is not simple data, it is semantic data, which makes it easier to query, browse and combine it. Apart from semantic data, it is also important a user interface that carries all this potential to the user. Rhizomer is a framework for semantic data publishing and user interaction that facilitates building semantic dashboards. It is possible, for instance, to build a simple dashboard on top of semantic data generated from financial reports and incorporate web services that provide specialised ways to interact with semantic data, like showing geo-located resources in a map or events in a timeline.

Keywords: Dashboard; Decision Support; Financial Data; Interaction; RDF; Semantic; Semantic Web

1 Introduction

One of the challenges of Decision Support Systems (DSS) in the context of increasingly virtual and distributed enterprises is data integration. Decision must be drawn from the combination of heterogeneous data coming from distributed and third party sources, whether it comes from structured relational databases or semi-structured content sources like documents or web pages. It's from the combination of all these sources that it is possible to attain unprecedented levels of information access, sharing and collaboration.

However, most current business intelligence implementations lack a technological foundation that facilitates the integration of data coming from a broader non-relational domain of data, which additionally might be distributed and outside enterprise boundaries and control. Semantic web technologies can help here, thanks to their ability to model and interlink data.

One additional benefit is that once in the Semantic Web, data is formalised in a way that logical inference can operate on it. Inference is guided by the knowledge captured by ontologies, which can model enterprise structure, business rules, etc. Moreover, web ontologies can be also distributed so it is easier to reuse both data and ontologies.

The original objective of the semantic web is to enable the description of web content using domain-specific data models (called ontologies) that make it possible for applications to locate and reason over the Web's resources. Similarly, ontologies can feasibly be adapted to model the structure of a data warehouse's schema.

It allows decision support systems queries to make semantic inferences over an extended range of external and distributed data that is not necessarily stored in the data warehouse; rather it is referenced through an ontology. The analysis is also enriched by a deeper semantic understanding of data relationships within the data warehouse itself, and is not restricted to the rigid relationships pre-coded in DB schemas.

Information often lacks a meaningful context. The knowledge extracted from unstructured sources must be enriched with links to ontologies that capture their context and facilitate then their integration and exploitation together with other more structured sources. There is some early work being done on semantically driven data integration with the semantic web equivalent of the SQL query language for accessing relational databases, SPARQL (Prud'hommeaux & Seaborne, 2008).

But semantic data is not enough if decision-makers cannot easily access and manipulate it. Semantic Dashboards challenge information and interaction design as well as information architecture. These

challenges come from integrating data from different sources in an open environment like that made possible by the Semantic Web and that enables entities like virtual world wide enterprises. Creating a dashboard architecture as a set of independent components whose configuration is driven by the available data.

This paper presents a the Rhizomer[1] framework for semantic data publishing and user interaction that can be used in order to develop decision support systems on top of semantic data. The platform is presented in Section 2. Then, in Section 3, a use scenario is presented. In that scenario, Rhizomer is used in order to publish, mix, query and browse semantic data resulting from the SEC's EDGAR program for financial reporting based on the XBRL standard.

In that scenario it is illustrated how, thanks to semantic technologies, it is possible to present and integrated view on XBRL plus data coming from other sources. On top of this integrated view it is possible to offer some basic user tasks that allow them interacting with data and make decision based on a more integrated and flexible data set. The current set of simple user task are the basis for future developments, in this and other application contexts, based on Rhizomer as a framework for semantic decision support systems.

1.1 Introduction to the Semantic Web

Looking at Semantic Web technologies and methodologies more relevant from the point of view of decision support systems, it is important to first note that it is rooted on a data model, a way to represent data, geared towards interoperability. It is based on a directed graph, i.e. a set of nodes connected by edges with a direction, from node A to node B. This graph model constitutes the first building block for semantic interoperability because a graph can be used to represent many other kinds of data structures.

For instance, it is easy to model a tree using a graph—it is just a graph without cycles or a table—each row is represented by a node that is connected to the different row values by edges labelled after each column name. This makes it easier to integrate data coming from XML documents or relational databases into the Semantic Web. Moreover, it is easier to mash-up data from disparate sources into a graph because the result is always a graph.

The Semantic Web graph model is named RDF, Resource Description Framework (Tauberer, 2008). However, this is not enough. We can put it all into a graph but, how do we tell the computer that one part of the graph can be joined to another part because they refer to the same thing? And, what is even more important, how do we put restrictions on how the graph is built in order to make it model interesting things and avoid that it becomes a messy bunch of nodes?

It is possible to accomplish these features using schemas and ontologies, at the Semantic Web and Ontological Engineering layer. First of all, they guide graph construction by providing restrictions on how nodes are connected to other nodes using different kinds of edges, called properties. For instance, it is possible to say that a node represents a person and that it is related through properties called "name", "e-mail" or "friend" to nodes providing the corresponding values for them.

RDF Schema is the simplest tool that allows modelling these restrictions (Daconta, Obrst, & Smith, 2003). It provides primitives similar to those from object oriented programming so it is possible to define classes with defined sets of properties and appropriate values. Classes are then used to categorise the things represented by nodes, called the resources, in order to apply the corresponding restrictions to them.

For instance, there is a class "Person", associated to the relevant properties for persons, which is applied to a node representing a given person. From this point, it is possible to guide how that person is described by the graph and, more importantly, the computer can interpret a description for that resource following the guidelines provided by the "Person" class.

Ontologies also provide ways to restrict how the graph is modelled, and how it should be interpreted by the computer (Fensel, 2004). They are a more sophisticated way to do so and are based on logic formalisms. This makes it possible to use logic reasoners in order to deduce new things about the data being managed. These kinds of deductions are a key feature in order to enable scalable data integration by computerised means. Computers use the clues and rules captured by ontologies in order to make sophisticated data integration at the semantic level, such as realising that two pieces of data match together or the kind of product that an invoice is referring to, e.g. from what the ontology says about the invoice, the customer, etc.

The Web Ontology Language (OWL) is used in order to define Semantic Web ontologies (Lacy, 2005). There are three sublanguages with different levels of complexity, which require increasing computation power but provide more expressive ways to pose restrictions. The simpler is OWL Lite and the more complex and expressive OWL Full. In the middle there is OWL DL, which is based on Description Logics (Baader, Calvanese, McGuinness, Nardi, & Patel-Schneider, 2003) and provides a trade-off between complexity and expressiveness. Ontologies provide the basis where semantic

processing and sophisticated semantic databases can be built, at the Semantic Web Based Information Systems Layer.

Finally, another relevant piece of the Semantic Web from the point of view of decision support systems is SPARQL (Prud'hommeaux & Seaborne, 2008). It is the Semantic Web query language, like SQL is the relational database query language. SPARQL allows building patterns to retrieve parts of a Semantic Web graph based on different constraints. This is going to be the piece on top of which decision support systems will be built in order to interact with the underlying Semantic Web model.

1.2 Related Work

Semantic Web technologies are starting to receive attention from the DSS community (Bose & Sugumaran, 2007). However, there are still very few developments that take profit from them. One example is the proposed Semantic DSS for the palm oil industry (Sabestinal, 2009). It takes profit from Semantic Web technologies in order to facilitate the fusion of heterogeneous information from distributed data sources. The approach is based on integrated usage of Semantic Technology through Ontologies, Context- Management and Web-services.

Another example for a different domain, concretely in the news domain, is Hermes (Borsje, Levering, & Frasincar, 2008). It is a framework that provides decision makers with the ability to extract a set of news items related to specific concepts of interest. This is accomplished by creating a knowledge base, based on Semantic Web technologies, and developing a system that classifies news with respect to the knowledge base.

However, these are all developments conceived for specific domain, i.e. DSS that are developed in order to cope with the requirements of a concrete problem. It is harder to find tools, based on Semantic Web technologies, intended for semantic DSS development from a generic stand.

A more generic approach is the Semantic Web Framework for Knowledge-Centric Clinical Decision Support Systems (Hussain, Raza Abidi, & Raza Abidi, 2007). This is a Semantic Web framework to both model and execute the knowledge within a Clinical Practice Guideline to develop knowledge-centric clinical DSS. The proposed approach entails knowledge modelling through a synergy between multiple ontologies, though it is also constrained to a concrete range of scenarios in the healthcare domain.

A really generic solution is BioDASH (Neumann & Quan, 2006). It is a Semantic Web prototype of a Drug Development Dashboard that associates disease, compounds, drug progression stages, molecular biology, and pathway knowledge for a team of users. Since such relevant information usually resides across many intranet database servers and different R&D groups, the challenge is more about leveraging the information one already has in a semantically coherent fashion than about making new data models.

Though BioDASH might seem another application in a specific domain, it is based on an extensible Semantic Web browser, Haystack (Quan, Huynh, & Karger, 2003). Consequently, it is possible to use this generic and extensible Semantic Web browser in order to develop semantic dashboards for DSS in other application domains. The main limitation of this tool is that it is not Web based. It is an application build on top of the Eclipse[2] framework. Consequently, it might become too heavyweight for wide distribution and difficult to maintain in a distributed way.

A pure-Web solution is more suited for distributed scenarios, where many different users get access to the tool from different locations. In this direction it has been recently announce Paggr[3]. It is a framework for semantic dashboards development based on Semantic Web technologies and PHP. Consequently, it is possible to develop Web applications that offer semantic dashboard functionality. However, Paggr is still under development so it has not been possible to evaluate it with detail. In any case, it is work that seems to go along the same path than our proposal for dashboards development for semantic DSS. Our proposal is detailed next.

2 The Rhizomer Platform

Rhizomer is a platform for semantic data publishing that provides mechanisms that facilitate user interaction with the published data. The user tasks Rhizomer gives support to are searching, browsing, sharing, annotating, mashing up and transacting. This platform can be used to develop semantic dashboards for decision support systems based on Semantic Web technologies.

In this context, the previous user tasks can be seen for instance as performing queries in order to retrieve relevant facts, browsing query results in order to contextualise the results or reach related data, sharing content or results that might be interesting for other users, attaching semantic descriptions to shared content or relevant facts in order to facilitate its management, mixing results from different

sources in order to get more than their simple addition and be able to draw more informed decisions and finally, to support user actions that cause changes in external systems.

From the technological point of view, Rhizomer is based on simple foundations, which make it flexible, scalable and capable of adapting to different deployment and use scenarios. Its core is rooted in simple HTTP mechanisms and follows a REST approach (Richardson & Ruby, 2007).

Each content item managed by the Rhizomer has its own URL, thus basing the whole system on a Resource Oriented Approach. The basic HTTP commands allow managing each resource. GET retrieves the resource description, PUT updates the specified resource with the provided description, POST creates a new resource generating its corresponding URL and DELETE removes the specified resource.

Additionally, the GET command is also used to pose semantic queries based on the SPARQL query language (Prud'hommeaux & Seaborne, 2008). The particularities of this language are hidden from the user, who builds queries interactively through semantic query forms as detailed in Section 2.1. These queries make it possible to retrieve the descriptions for resources satisfying the constraints defined by the query.

All the previous HTTP functionality is implemented by the Rhizomer server component, shown in Figure 1. This component lies on top of a semantic metadata repository, like any one supporting the Jena API, and implements the previous HTTP commands in order to facilitate the manipulation of semantic metadata.

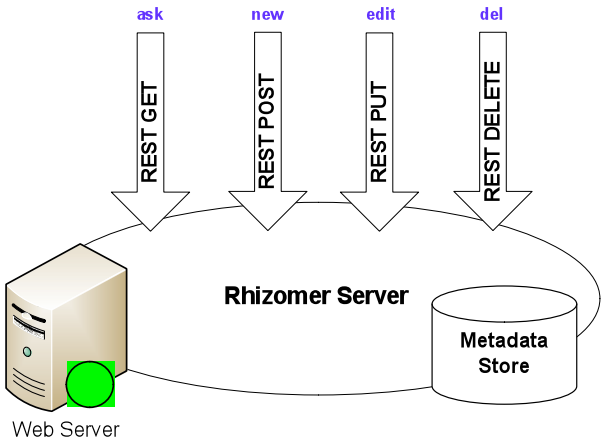

Figure 1. The Rhizomer Server architecture

On the other hand, the client-part functionalities have been developed with the aim of improving the usability of the user interface. They are encapsulated in the user browser and implemented using Javascript and asynchronous HTTP calls (AJAX) (Crane, Pascarello, & James, 2005).

The whole user experience is built on top of these operations. However, the RDF syntax of semantic metadata is completely hidden in order to increase usability. As end-users typically interact with HTML web pages through their browsers, Rhizomer incorporates a generic transformation from RDF to HTML. This transformation is not tied to any particular ontology or scheme, as it is the case in most template-based approaches like Fresnel lenses (Pietriga, Bizer, Karger, & Lee, 2006). This HTML rendering is used to build a transparent browsing experience on top of the SPARQL endpoint that retrieves semantic metadata.

The browsing steps are based on a fragmentation of the underlying RDF graph, which is detailed in Section 2.2. The same fragments are used in order to constraint the range of the update and deletion operations, as it is detailed in Section 2.3. Updates and new metadata generation are carried out through semantic-enabled HTML forms, which also help to hide the burdens of RDF metadata from users.

Besides, Rhizomer incorporates Semantic Web services providing interaction means beyond content retrieval and metadata browsing and search. Moreover, the integrated services can be used in order to perform transaction with external systems independent from Rhizomer. Each user action corresponds to a Semantic Web service whose description incorporates the constraints a resource must satisfy to be a valid input for the service. Consequently, the semantic description of a resource determines which actions can be applied to it.

For instance, let us consider a scenario where the platform is used to retrieve and browse a set of news items described using semantic metadata. These descriptions include date and time information and, in some cases, the geographical location of the event covered by the news items. At first, these descriptions are visualised as generic HTML pages based on the RDF to HTML rendering. These pages

allow the user to visualise the descriptions of the corresponding resources, the metadata, and to browse them interactively by navigating the underlying graph.

This generic approach can be applied to any kind of resource. However, in this scenario, it would surely be desirable to have more specific views and more accurate ways to interact with events. Calendars or timelines are good choices for time stamped resources, while maps should be helpful for geographically located ones.

The objective of the Rhizomer platform, and the reason why Semantic Web services have been chosen as the way to implement actions, is to build a generic and dynamic system which can directly deal with generic RDF metadata while being easily extensible to incorporate specialised ways to view and interact with particular kinds of resources. There are more details about this in Section 2.4.

2.1 Search

First of all, Rhizomer allows users to perform semantic queries without any knowledge of semantic query languages. All they need is to know how to fill query forms. These forms are generated dynamically from the kind of resource they are interested in, more concretely from the properties specific for that kind of resource.

The kind of resources the users is interested in will usually be selected from the global navigation bar that usually appears at the top part of the page. This bar is currently generated statically. The site deployer defines which these main categories are when the site is first configured. This kind of resources, which corresponds to a class of one of the ontologies used by the site running Rhizomer, is used in order to query for all the properties whose domain is that class or any superclass of it, plus all the properties for which there is a restriction when the property applies to that class or its superclasses, e.g. the ontology statement *SubClassOf(gr:Offering AllValuesFrom(gr:includesObject gr:Object))* is used in order to retrieve the "includesObject" property as one applying to the class "Offering".

The corresponding SPARQL query template is shown in Table 1. This query allows dynamically retrieving the properties specific to a kind of resource. The list is then used in order to generate a form with one input field for each property. The user can then fill the form in order to establish the search criteria and when the form is submitted the corresponding SPARQL query will be generated. That query will retrieve all resources with the properties that have been filled in the form and whose values contain the input field filler.

Table 1. SPARQL query that retrieves the properties specific for a class

```
SELECT ?p
WHERE
{
    {   ?p rdfs:domain ?d.
        ?t rdfs:subClassOf ?d.
        FILTER (?t = [CLASS] && ?d != rdfs:Resource) }
    UNION
    {   ?r rdf:type owl:Restriction.
        ?r owl:onProperty ?p.
        ?t rdfs:subClassOf ?r.
        FILTER (?t = [CLASS]) }
}
```

Moreover, users can add other properties that without being specific, might also apply to resources of that kind. These properties are also dynamically retrieved using the SPARQL query shown in Table 2. This query retrieves all properties (RDF properties or OWL data-type or object-type properties) that are generically defined as having any resource as domain or that do not have any domain defined.

Table 2. SPARQL query that retrieves the properties specific for a class

```
SELECT ?p
WHERE
{
    ?p rdf:type ?t.
    FILTER (?t = rdf:Property || ?t = owl:DatatypeProperty || ?t=owl:ObjectProperty).
    OPTIONAL
        { ?p rdfs:domain ?d }.
    FILTER (?d=rdfs:Resource || ! bound(?d))
}
```

Query results are those resources satisfying the search criteria. However, result pages show more information than just the identifiers of the retrieved resources. To provide more context to the user, each

resource is presented together with its description. The user does not get these descriptions as raw data but an HTML rendering of the descriptions where all bad-looking URLs are substituted by more readable labels. More details about the HTML rendering of the retrieved semantic metadata are presented in the next subsection.

2.2 Metadata Browsing

Browsing is the basic interaction paradigm in the Web. It is based on the successive visualisation of Web pages following the links connecting them. Pages and links are the main building blocks upon which the interaction is built. Web pages are intended for human users' consumption and well-established methodologies to make them usable and accessible exist.

However, neither the browsing paradigm nor the principles to make the whole thing usable and accessible can be directly applied to the Semantic Web. That is so because it is based on a model not built upon pages and links but on triples (subject-predicate-object), which makes the browsing approach quite different from the Web.

The combination of many triples builds up a graph and, though the resulting model is easier to process by computers, the consumers of Semantic Web metadata are, at the end, humans so usable and accessible interaction mechanisms are required.

First of all, the basic browsing paradigm should change because the Semantic Web makes it very difficult to base the browsing steps on documents. In other words, it does not seem appropriate, for each step, to show all the triples in the corresponding document to the user as it is done in the Web. The amount of information in a single document can be too large, more than thousands of triples. Moreover, the frontiers among documents are very fuzzy: usually, many documents are combined in order to get a coherent graph.

Semantic Web browsers like Tabulator (Berners-Lee, Hollenbach, Lu, Presbrey, Prud'hommeaux, & Schraefel, 2007) follow this approach and show all the triples from a Semantic Web document as an unfoldable tree. As preliminary user tests show, this approach causes many usability problems because, as the tree grows, it rapidly becomes difficult to manage. As it has been said, documents contain many triples and, additionally, each navigation step adds more triples from the new document to the current set.

Another approach is faceted browsing, as in /facet (Hildebrand, Ossenbruggen, & Hardman, 2006). However, our objective is a simpler and more polyvalent browsing mechanism, which deals better with heterogeneous information spaces, although it might lack the guidance provided by facets. Anyway, it is not clear how systems like /facet deal with metadata structures that feature many anonymous resources, as it is the case for the semantic metadata managed in the application scenario described in Section 3.

Therefore, the problem is where to put the limits of each browsing step when presenting semantic metadata. In other words, how each browsing piece is built and how new pieces are created and presented following user needs in order to compose a browsing experience through the whole graph.

In order to facilitate browsing, the proposed approach is based on the construction of graph fragments. Following this approach, it is possible to construct fragments for any graph starting from any node that is not anonymous. For instance, the starting point for the metadata describing a piece of content is the node that represents the piece, which is the subject for all the triples describing it. This node has an ID and consequently is not anonymous.

All the triples that start from this node are part of the fragment and the triples describing anonymous objects are also added to the fragment. This happens for all nodes that are only identifiable in the context of the starting node. For instance, Figure 3 shows how an example graph would be fragmented following this approach.

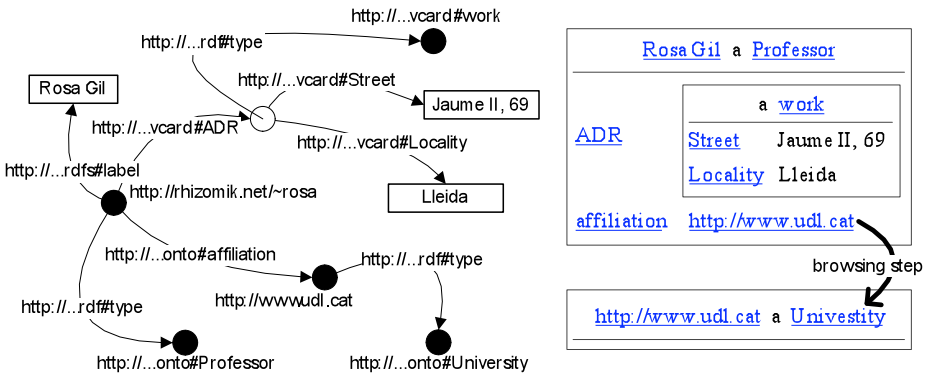

Figure 3. Fragmentation of an example RDF graph and the resulting HTML rendering

As it can be seen, there are two fragments, each one corresponding to an identified resource described by at least one triple, for which it is the subject. The first fragment describes http://rhizomik.net/~rosa and includes an anonymous resource for the address. The second one, for http://www.udl.cat, can be reached from the first one through a browsing step. Unlike the address, it is shown independently because it is not anonymous.

This process of building fragments continues iteratively and interactively because from a fragment, if the user wants more information about a resource, it is possible to follow the next browsing step starting from any non anonymous object: the metadata describing the selected identified node is retrieved and the new fragment is built. Moreover, when the results are rendered to the users all the URIs are replaced by labels, if they are available, making the results much more usable.

Finally, fragments are rendered using HTML, which is viewable using a web browser, a tool users feel comfortable with. In order to generate HTML from RDF, fragments are serialised as RDF/XML and transformed using an XSL. The XSL transformation, which is part of the Rhizomer platform, guarantees consistent results whenever the input RDF/XML has been generated from fragments based on the Rhizomer approach.

This mechanism has been implemented as successive DESCRIBE queries for the identified resource URIs to the SPARQL endpoint. The DESCRIBE operation of the SPARQL endpoint has been re-implemented in order to build the proposed fragments, which also include all the available labels. Labels are used, when available, in the place of resources URLs in order to make the HTML rendering more readable.

Then, the XSL transformation from RDF/XML to HTML is invoked from the client using AJAX, which is also responsible for sending the SPARQL queries and making the whole process go smoothly behind the scenes, making the user experience even more comfortable. In practice, Rhizomer becomes a Semantic Web browser, which can even carry the user to resources whose descriptions are stored outside Rhizomer making it possible to easily reuse external metadata. For instance, it is possible to point to resources in DBPedia and show their descriptions using Rhizomer when the users' clicks on them.

Finally, the AJAX part of Rhizomer at the client also keeps track of the browsing steps so it is possible to use the "back" and "forward" browser buttons. Moreover, the browsing steps are cached at the browser in order to improve responsiveness.

2.3 Editing Metadata

The previous fragment-based approach, besides being the foundation for browsing, allows constraining the metadata editing and deletion actions to a limited set of triples. This way, it is possible to implement editing actions as the replacement of a given fragment with the one resulting from the editing process. The same strategy applies for the deletion action.

All these operations are also carried out through an HTML interface. In addition to the RDF to HTML transformation, the Rhizomer platform includes an XSL transformation from RDF to HTML forms. These forms are generated automatically from the RDF/XML corresponding to a fragment. The same approach as in the RDF to HTML transformation is followed but, instead of generating text values and links for literals and resource, this transformation generates input fields for each triple. The field is named using the corresponding property URI and its value corresponds to the triple value. The fields can be used in order to edit the property value, either a resource URIs or a literal.

Moreover, properties and values can be removed or added. Currently, the user enjoys some assistance during the editing process. Basically, when the user chooses to add a new property, a SPARQL query is used in order to retrieve all the available properties for the resource being edited. These are the properties not constrained to a particular resource type plus those constrained to the types of the resource being edited. In order to do that, we use the same queries than when building query forms. However, in this case, we combine the properties for all the classes the edited resource is an instance of.

The future plan is to improve this support in order to assist users also when they provide values for the selected properties. Currently, they just have an input field where they can type the data-type value for attributes or the URI for relations. The idea is to also use SPARQL queries against all the available values in the repository in order to provide some recommended values.

Finally, an algorithm has been developed in order to reverse the mapping from RDF to HTML forms. In other words, this algorithm is responsible for generating the RDF that results from the editing process by mapping the form input fields to the corresponding triples. There is an input field, generated during the RDF to HTML form step, which stores the URI of the resource being edited. This one becomes the

subject for all the RDF statements (triples) generated from that form. The input field identifiers and their fillers become the subjects and objects for that triples describing the subject resource.

It is even possible to edit descriptions that point to anonymous resources. In these cases there are hidden input fields that keep track of what subject each input field "property" is describing. Figure 5 shows a graphical representation of this process.

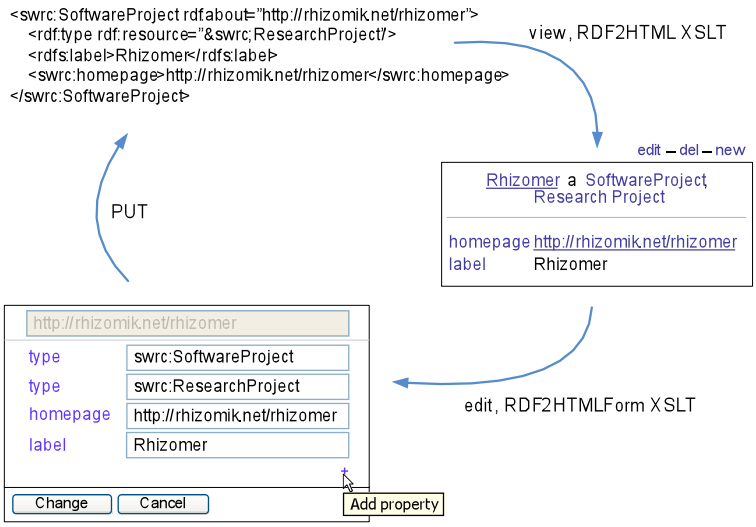

Figure 5. Transformation from RDF to HTML form and back to edited RDF

2.4 Actions as Semantic Web Services

The metadata browsing and editing components presented in the previous sections give users access to resources and their descriptions. The user can pose queries to access the descriptions of the resources managed by the system, browse through the semantic metadata that describe them or edit these descriptions.

In addition to the previous user actions, once the object (or objects) of interest is located, the specific actions that the user can do upon it are shown. In the Rhizomer platform, this part is implemented by means of semantic web services. This allows a completely dynamic integration of the actions because they are not predefined for the different types of objects, i.e. they can be seen as independent entities.

The actions supported do not require complex web services so the actions in Rhizomer are implemented as web services based on REST. That is to say: simple HTTP requests to the services that get HTTP responses with the result. For instance, Yahoo! Maps provides a REST interface to a service that, given the geographical coordinates to show, returns its location in a map.

REST simplifies the invocation of web services and it is only concerned with this aspect. Therefore, for the localization and automatic invocation of REST-based web services, formal descriptions of these services are needed. We believe that the initiatives of semantic web services are the answer to this problems and that is why we have considered the modelling mechanisms they provide.

It has been considered that the ontologies provided by OWL-S 1.1 (Martin, 2004) are the most appropriate for describing our web services due to their modularity. It has been easier to detect the classes and properties more appropriate to the kind of descriptions we require and use them in isolation without any concern about the rest of the framework. Only the *Service Profile* provided by OWL-S is used for a high-level description of the service. Neither *Service Grounding* nor *Service Model* are considered because the simplicity of the REST services considered do not make them necessary.

In fact, in the current state of the system, only the class *Process* and the properties *hasInput* and *hasOutput* (defined in OWL-S) are used. *Process* allows identifying the resources that correspond to web services that can be invoked from Rhizomer. Their URIs correspond to the service's access point, so it must be an URL. Input parameters for the service are not used but data is sent in the body of a POST message and corresponds to the RDF/XML serialisation of the description of the resource (or resources) that the service accepts as input.

The *hasInput* property is associated to *Process* resources and identifies the class of things that serves as input for the service. Consequently, for a service to appear as available when a concrete resource is shown, this resource must belong to the class defined as the input of the service. It is not necessary to make an *a priori* classification of the resource. To get the desired dynamism, classes in OWL can be

specified to be used in *hasInput* that represent the necessary and sufficient conditions to classify resources automatically. This is possible with a Description Logic (DL) reasoner.

For instance, as it is shown in Table 3, it is possible to define *GeolocatedEntity* as the class of all the resources with properties *lat* and *long* and use it as the *hasInput* class for a service named "map". There is no need to explicitly classify all the geolocated entities into this class. The reasoner is responsible for classifying into it all the resources that satisfy these restrictions.

Table 3. Description of a geographical information visualization service

```
<rdf:RDF ...
  xmlns:process="…/services/owl-s/1.1/Process.owl#"
  xmlns:pos="…w3.org/2003/01/geo/wgs84_pos#">
<process:Process
        rdf:about="http://rhizomik.net/rhizomer/services/map">
    <rdfs:label>map</rdfs:label>
    <process:hasInput>
        <owl:Class rdf:ID="GeolocatedEntity">
            <owl:intersectionOf rdf:parseType="Collection">
            <owl:Restriction>
                <owl:onProperty rdf:resource="&pos;lat"/>
                <owl:minCardinality>1</owl:minCardinality>
            </owl:Restriction>
            <owl:Restriction>
                <owl:onProperty rdf:resource="&pos;long"/>
                <owl:minCardinality>1</owl:minCardinality>
                </owl:Restriction>
            </owl:intersectionOf>
        </owl:Class>
    </process:hasInput>
    <process:hasOutput>text/html</process:hasOutput>
</process:Process></rdf:RDF>
```

Then, when the user is browsing resource descriptions, it is checked whether they correspond with the input class of any of the available services. For instance, when a resource has both latitude and longitude, the reasoner classifies it as an instance of *GeolocatedEntity*, so it is detected as being accepted by the "map" service. Consequently, this service can be invoked passing a description of the corresponding resource as its input. The user can invoke the service using a link, automatically associated to the resource using the mechanism described before, and get a visualization of the position of the resource in a map.

Direct invocation of web services passing them the RDF metadata of the resource that must be used as input is not usually allowed. Therefore, in many cases, the URL associated with a service is actually pointing to a wrapper that receives the RDF, extracts the data needed by the service, and makes the "real" invocation of the service. This additional layer between Rhizomer and the services, though it complicates the implementation, allows using visualisation services such as GoogleMaps or SIMILE Timeline[4] that are only available as JavaScript libraries. In this case the wrapper is implemented as a servlet that generates the web page that uses the JavaScript library and provides the final result.

Finally, the *hasOutput* property specifies the output type of the service. For visualization services a literal representing the MIME type of the output is used. The output is shown in a new HTML layer within the Rhizomer interface and the MIME type is used to correctly interpreting the result.

3 Semantic Financial Reports Scenario

This section illustrates how Semantic Web technologies and methodologies can be used in order to enrich and integrate existing data. Then, once that data has been transformed into semantic data, it is shown how the Rhizomer platform can be applied in order to build a semantic dashboard allows users to interact with that data.

In this case, the existing non-semantic data is XBRL data. XBRL (eXtensible Business Reporting Language) is an XML language intended for modelling, exchanging and automatically processing business and financial information. XBRL is starting to be deployed in many different scenarios. For instance, there is the EDGAR[5] program promoted by the U.S. Securities and Exchange Commission (SEC).

It performs automated collection, validation, indexing, acceptance and forwarding of submissions by companies and others who are required by law to file forms with the SEC. Currently, filers may choose to voluntarily submit documents in XBRL format to accompany certain official filings. However, the SEC

mandate is that in the next years all enterprise obliged to report on their business data will do so using XBRL.

However, we have observed limited support for cross analysis of financial information in XBRL tools and applications (García & Gil, 2009). This is not just among data based on different accounting principles, which are represented in XBRL using taxonomies. It even happens when comparing filings for different companies based on the same taxonomies or filings for the same company based on different versions of the taxonomies. This makes it difficult to build tools that offer an integrated view on all this data to users and facilitate the decision process.

We argue that this limitation is inherited from the technologies underlying XBRL, especially XML. XML takes a document-oriented approach, where each document presents a tree structure. This makes it difficult for XML-based tools to provide functionalities that blur this separation into documents and that overcome the limitations of a tree structure when mashing-up data from different sources. Moreover, XBRL does not provide formal semantics that might help to integrate different taxonomies by using logic reasoners.

In any case, the integration of data contained in XBRL into comparable information is a strong requirement for the analysis of business and financial information at the global level. This might increase the efficiency and effectiveness of the decision making processes relying on this kind of information. For instance, bankruptcy prediction and other tasks related to the assessment of the solvency of a firm, a business sector or set of interrelated companies.

Many have already pointed to this issue and proposed Semantic Web technologies as a natural choice for XBRL data integration (Hoffman, 2006). However, despite these benefits, currently, financial and business data is being produced using XBRL and it seems that more and more XBRL data is going to be available in the future. It is been promoted by regulators and government agencies like the SEC and other entities like the European Union or the Spanish securities commission (Núñez, de Andrés, Gayo, & Ordoñez, 2008).

Consequently, we think that the best approach in order to get financial and business data to the Semantic Web is not to propose an alternative language based on Semantic Web technologies, but to apply methods to map existing XBRL to semantic metadata. This approach, its results and its validations are presented in the following sections, after XBRL is introduced.

3.1 Approach

In order to move existing XBRL instances and taxonomies to the Semantic Web, and due to the fact that XBRL is based on XML and XML Schema, we have applied the XML Semantics Reuse methodology (García, 2006). This methodology is implemented as two mappings by the ReDeFer project[6], the first one from XML Schema to OWL and the second one from XML to RDF.

This approach has already shown its usefulness with other quite big XML Schemas, especially in the multimedia metadata domain (García, Perdrix, Gil, & Oliva, 2008), where it has produced the more complete MPEG-7 ontology to date (García, Tsinaraki, Celma, & Christodoulakis, 2008).

3.2 XSD2OWL Mapping

The XML Schema to OWL mapping is responsible for capturing the schema semantics. This semantics are determined by the combination of XML Schema constructs. The mapping is based on translating these constructs to the OWL ones that best capture their intended meaning. These translations are detailed in Table 5.

Table 5. XSD2OWL translations for the XML Schema constructs and shared semantics with OWL constructs

XML Schema	OWL	Mapping motivation
element | attribute	rdf:Property owl:DatatypeProperty owl:ObjectProperty	Named relation between nodes or nodes and values
element@substitutionGroup	rdfs:subPropertyOf	Relation can appear in place of a more general one
element@type	rdfs:range	The relation range kind
complexType|group | attributeGroup	owl:Class	Relations and contextual restrictions package
complexType//element	owl:Restriction	Contextualised restriction of a relation

extension@base | restriction@base	rdfs:subClassOf	Package concretises the base package
@maxOccurs @minOccurs	owl:maxCardinality owl:minCardinality	Restrict the number of occurrences of a relation

3.3 XML2RDF Mapping

Once all the metadata XML Schemas are available as mapped OWL ontologies, it is time to map the XML metadata that instantiates them. The mapping is based on modeling the XML structure, i.e. a tree, using RDF.

The fundamental translation is between relations, from *xsd:elements* and *xsd:attributes* to *rdf:Properties*. Concretely, *owl:ObjectProperties* for node to node relations and *owl:DatatypeProperties* for node to value ones.

Values are kept during the translation as simple types and RDF blank nodes are introduced in the RDF model in order to serve as the source and destination for properties. They will remain blank for the moment until they are enriched with semantic information.

The resulting RDF graph model contains all that we can obtain from the XML tree. It is already semantically enriched thanks to the *rdf:type* relation that connects each RDF property to the *owl:ObjectProperty* or *owl:DatatypeProperty* it instantiates. It can be enriched further if the blank nodes are related to the *owl:Class* that defines the package of properties and associated restrictions they contain, i.e. the corresponding *xsd:complexType*. This semantic decoration of the graph is formalised using *rdf:type* relations from blank nodes to the corresponding OWL classes.

3.4 Results

First of all, we have generated an ontological infrastructure for the XBRL core, currently XBRL 2.1. It is composed by the ontologies resulting from mapping the XBRL core XML Schemas using the XSD2OWL mapping: XBRL Instance, XBRL Linkbase, XBRL XL and XBRL XLink. Apart from the previous schemas, the EDGAR Standard Taxonomies schemas have been also mapped in order to be able to map the XBRL data submitted to the XBRL voluntary program EDGAR.

Each filing for the companies participating in the EDGAR program contains an XBRL XML file representing the actual financial data and also a specific XML Schema extending the XBRL core. This schema provides specific guides for the corresponding financial data. Both files are mapped using XML2RDF and XSD2OWL respectively.

For instance, for Adobe Systems Inc filing on 2008-07-03, there are the adbe-20080616.xml file containing the instance data and the adbe-20080530.xsd schema for data structures specific for this filing. They are mapped, respectively; to the RDF file for instance data adbe-20080616.rdf and the OWL ontology adbe-20080530.owl for the schema.

All the previous ontologies are available from the BizOntos Business Ontologies web page[7] and the semantic data for all the processed filings can be queried and browsed from the Semantic XBRL site[8]. Currently, 489 filings have been processed from EDGAR. The combination of all these filings once mapped to RDF amounts slightly more than 1 million triples, concretely 1,023,929 triples.

Finally, the generated data is published as Linked Open Data in the World Wide Web. The approach is based on generating XHTML plus RDFa (Halb, Raimond, & Hausenblas, 2008). In order to do that, we have used the Rhizomer platform that, apart from encapsulating the metadata store, also provides an RDF to XHTML+RDFa transformation and a RDF to HTML Form transformation that makes it possible for users to interactively edit the published data. The whole architecture is shown in Figure 4, which apart from the semantic data generation and publishing functionalities also features links to external sources of information as detailed in the next section.

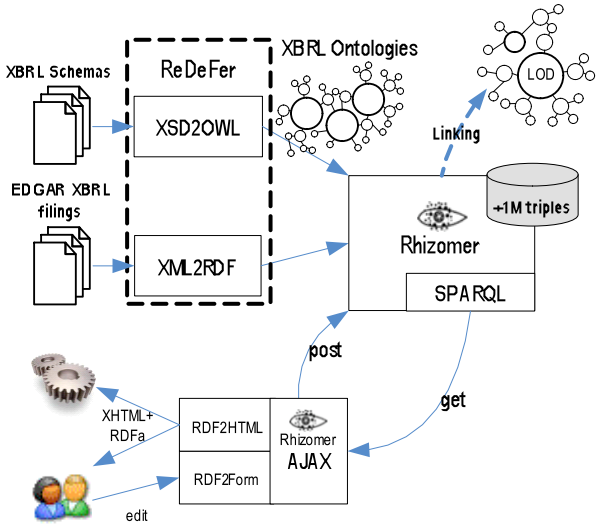

Figure 7. Architecture of the proposed solution for semantic XBRL generation, linking and publishing

3.5 Integration of Additional Sources

One of the greatest benefits of Semantic Web technologies is that they facilitate integrating other sources of data in a process called data mash-up that can get out of the integration more that the sum of the parts, with the consequent increase of data value. This is especially true for semantic data made available using the principles of Linked Data (Berners-Lee, 2007):

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information.
4. Include links to other URIs so that they can discover more things.

Following these principles, it is possible to directly retrieve semantic metadata about resources just using HTTP. Available sources of semantic data that follow this approach are DBPedia (Auer, Bizer, Kobilarov, Lehmann, Cyganiak, & Ives, 2008) or Freebase[9]. From them it is possible to retrieve facts about places, people, companies, etc. that can be easily integrated with local data in order to enrich it.

Another important source of semantic data is also unstructured content, especially news feeds, that can be processed through Reuters's service Open Calais[10]. This service is capable of extracting references to named entities like companies or people from text, disambiguate them and provide links to DBPedia or Freebase for further information. It is also capable of extracting simple facts that involve those kinds of entities, like mergers, acquisitions, appointments, etc.

3.6 Examples of Use

Rhizomer is a framework for semantic data publishing and user interaction. It provides simple ways of interacting with data, on top of which it is possible to develop more sophisticated and scenario specific functionalities. In the case of the financial reports scenario, these simple services provided by Rhizomer have been used in order to implement a web dashboard[11] through with semantic financial reports can be searched, browsed and edited.

Users can search the whole set of reports using semantic queries based on SPARQL. It is possible to write those queries directly, using the advanced search page that features a form that allows writing down the SPARQL query, see Figure 8. This form is intended for users with knowledge about SPARQL, though the form features some query examples and it is possible to write and test queries in the same page in an interactive and quick way, results from the query are shown below it.

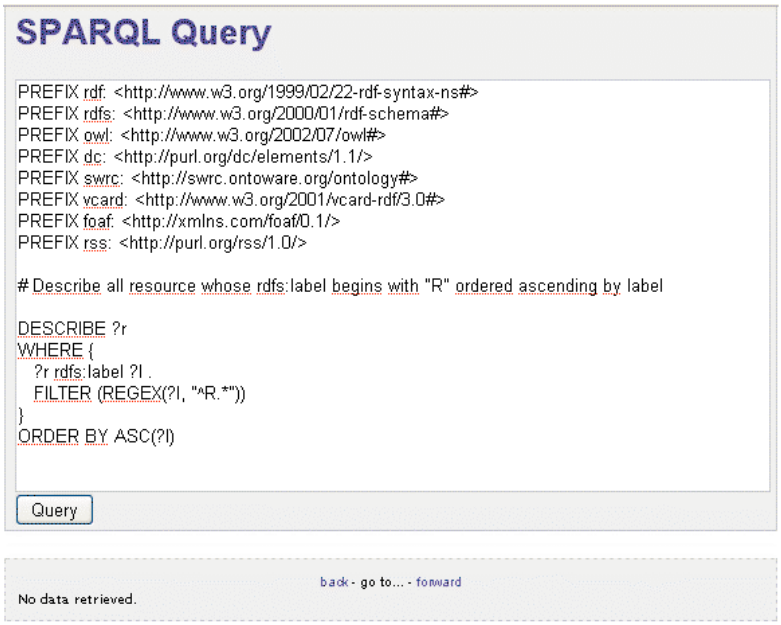

Figure 8. SPARQL query interface for advanced users

For users without knowledge about SPARQL, or that want to avoid the burden of writing these queries by hand, it is also possible to build them interactively by filling and adding fields to a form, as shown in Figure 10. In this case, the user selects the kind of resource she or he is interested in from the set of classes defined in the XBRL ontology, in this case the Context class. A basic form is automatically generated with fields for all those properties that are specific to this kind of resources. Additionally, the user can add more search criteria by adding additional properties from those defined in the XBRL ontology and constrain their values by filling the corresponding form field. Finally, when the user clicks the "Search" button, a SPARQL query is automatically generated for all resources with properties whose values are equal or similar to the corresponding field fillers.

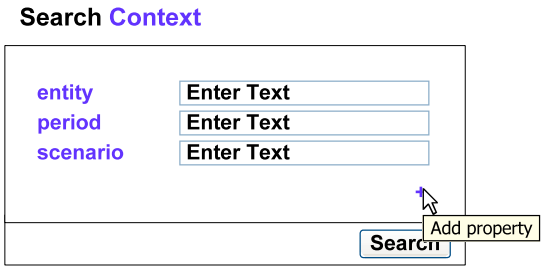

Figure 10. Dynamic form for semantic query building

Once the user has performed a query, results are presented as detailed in Section 2.2. They can also be edited as detailed in Section 2.3. One additional feature in this scenario is that some resources in the financial reports have been linked to more detailed descriptions of them available from DBPedia. One case of this are the companies mentioned in XBRL reports. Reports do not provide further information about the company that might help decision building beyond the financial data. Thanks to this links, and the ability of Rhizomer to browse semantic data following them, it is possible to transparently browse the information available from DBPedia for a company starting from some fact about it in one of the reports. For instance, Figure 12 shows the facts available from DBPedia for Adobe Systems.

Figure 12. Browsing information about a company from DBPedia

Finally, it is also possible to use the interaction services bundled with Rhizomer in order to interact with report data. There is not geographic information in XBRL reports but it is possible to get it from DBPedia. For instance, it is possible to browse for a financial fact to the company involved, then to data about it from DBPedia, which includes where its headquarters are located and display their location in a map. The other interaction service available by default is timeline. It is possible to show a set of financial facts in a timeline taking into account the "period" property defined in financial facts contexts.

This are just examples of use of the interaction services and functionalities bundled with Rhizomer by default and how they can be used in order to build a simple web dashboard that helps users interact with semantic financial data extracted from XBRL reports. More services might be easily integrated using the mechanisms provided by Rhizomer. Moreover, the whole dashboard can be easily customised by modifying the underlying templates and style sheets.

4 Conclusions and Future Work

Though there are some attempts to profit from Semantic Web technologies for DSS, as shown in the related work section, they are still quite preliminary. Most of them are very specific developments, taylored for concrete scenarios. There are even few attempts to build frameworks that facilitate building semantic dashboards for DSS, especially if the objective is to build a Web-based system.

Such kind of systems, Semantic Web-based dashboards for decision support, can benefit for the amount of semantic data available, still quite limited but quickly growing[12] in order to make possible more informed decisions at a lower cost. Semantic Web technologies facilitate mashing up local data with semantic data available for the Web of linked data.

Rhizomer constitutes a simple and flexible framework for semantic data publishing and user interaction. The platform offers a generic RDF to HTML transformation that makes it possible to navigate through semantic metadata and the associate ontologies. Apart from metadata browsing capabilities and content retrieval, users can carry out additional actions implemented by means of Semantic Web services.

These services are associated to the resources by a matching process based on their semantic descriptions. Currently, there are two services bundled with Rhizomer, a service that shows geo-located resources in a map and a service that shows resources in a time line. However, additional services can be easily and dynamically added as whole range of services provided by the platform is built on top of the semantic metadata and the associated ontologies.

It has been show how it is possible to use this set of simple publishing and interaction services provided by Rhizomer in order to build a semantic dashboard in a concrete scenario. The scenario is

about financial data available from XBRL filings. This kind of data is based on XML technologies but it is possible to map it to Semantic Web technologies. Then, it becomes easier to reuse semantic data from services like DBPedia in order to enrich the financial data from the reports.

For instance, reports are about companies but little information about them is available. However, it is possible to use company identifiers and other properties in order to integrate the financial data about a company with general information from DBPedia, e.g. descriptions, locations, related companies and people, etc.

End users can then benefit from Rhizomer in order to perform semantic queries about all this data, by directly writing SPARQL queries or assisted through configurable forms. They can also browse results, following all relationships among the resources and even arriving to the descriptions available from external sources like DBPedia or Freebase.

It is important to note that these sources are external to the local data directly managed by Rhizomer and its underlying semantic data store. Rhizomer behaves also as a Semantic Web browser that is capable to browse the semantic data available in other sources without the need of storing and maintaining it. However, this data also benefits from the interaction services provided by Rhizomer and, apart from browsing it, it is also possible to apply interaction services on them when appropriate. For instance, it is possible to browse to the semantic description of a company in DBPedia and then show its location in a map using this Rhizomer service.

Future work focuses on enlarging the range of interaction services directly available from Rhizomer for the financial data and other scenarios. The application scenarios will help testing available services and motivating new ones. In the short term, most efforts will concentrate on the financial data scenario as the amount of semantic data available is big once an automatic mapping from XBRL to Semantic Web data has been defined.

Moreover, another interesting point is to improve the metadata edition facilities beyond the assistance when a user tries to add a new property to the current description. Property ranges and restrictions that apply to the kind of resource being edited will be considered to suggest resources, which could constitute a proper value for the property. This will help managing the available data, which might come from many different sources.

Acknowledgments

The work described in this paper has been partially supported by Spanish Ministry of Science and Innovation through the Open Platform for Multichannel Content Distribution Management (OMediaDis) research project (TIN2008-06228).

References

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., & Ives, Z. (2008). DBpedia: A nucleus for a web of open data. In *The semantic web, ISWC/ASCW'07* (LNCS 4825, pp. 722-735).

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., & Patel-Schneider, P. (2003). *The description logic handbook: Theory, implementation and applications*. Cambridge, UK: Cambridge University Press.

Berners-Lee, T. (2007). *Linked data* (Design Issues Report).
Retrieved from http://www.w3.org/DesignIssues/LinkedData.html

Berners-Lee, T., Hollenbach, J., Lu, K., Presbrey, J., Prud'hommeaux, E., & Schraefel, M. C. (2007). *Tabulator redux: Writing into the semantic web* (Tech. Rep.). Southampton, UK: Electronics and Computer Science, University of Southampton. Retrieved from http://eprints.ecs.soton.ac.uk/14773

Bizer, C., Heath, T., Idehen, K., & Berners-Lee, T. (2008). *Linked data on the web (LDOW2008)*. In Proceedings of the 17th International WWW Conference (pp. 1265-1266). ACM Publishing.

Borsje, J., Levering, L., & Frasincar, F. (2008). Hermes: A semantic web-based news decision support system. In *Proceedings of the 2008 ACM Symposium on Applied Computing,* Fortaleza, Brazil (pp. 2415-2420). ACM Publishing.

Bose, R., & Sugumaran, V. (2007). Semantic web technologies for enhancing intelligent DSS environments. In *Decision support for global enterprises* (Vol. 2, pp. 221-238). New York: Springer.

Crane, D., Pascarello, E., & James, D. (2005). Ajax in Action. Greenwich, CO: Manning.

Daconta, M. C., Obrst, L. J., & Smith, K. T. (2003). *The semantic web: A guide to the future of XML, web services, and knowledge management*. Indianapolis, IN: Wiley.

Fensel, D. (2004). *Ontologies: A silver bullet for knowledge management and electronic commerce*. Heidelberg, Germany: Springer.

García, R. (2006). XML semantics reuse (PhD thesis, Universitat Pompeu Fabra). In *A semantic web approach to digital rights management* (Ch. 7). Retrieved from http://rhizomik.net/~roberto/thesis

García, R., & Gil, R. (2009). *Publishing XBRL as linked open data*. Paper presented at the Linked Open Data Workshop 2009, World Wide Web Conference, Madrid, Spain.

García, R., Perdrix, F., Gil, R., & Oliva, M. (2008). The semantic web as a newspaper media convergence facilitator. *Journal of Web Semantics, 6*(2), 151-161.

García, R., Tsinaraki, C., Celma, O., & Christodoulakis, S. (2008). Multimedia content description using semantic web languages. In Y. Kompatsiaris & P. Hobson (Eds.), *Semantic multimedia and ontologies: Theory and applications* (pp. 17-54). Springer.

Halb, W., Raimond, Y., & Hausenblas, M. (2008). Building Linked Data For Both Humans and Machines. In proceedings of the Linked Data on the Web Workshop (LDOW'08), Beijing, China.

Hildebrand, M., Ossenbruggen, J., & Hardman, L. (2006). /facet: A browser for heterogeneous semantic web repositories. In *Proceedings of the International Semantic Web Conference 2006* (LNCS 4273, pp. 272-285).

Hoffman, C. (2006). *Financial reporting using XBRL: IFRS and US GAAP edition*. Retrieved from http://www.lulu.com.

Hussain, S., Raza Abidi, S., & Raza Abidi, S. S. (2007). Semantic web framework for knowledge-centric clinical decision support systems. In *Artificial intelligence in medicine* (LNCS 4594, pp. 451-455).

Lacy, L. W. (2005). *OWL: Representing information using the web ontology language*. Bloomington, IN: Trafford Publishing.

Martin, D. (Ed.). (2004). *OWL-S: Semantic markup for web services*. Retrieved from http://www.w3.org/Submission/OWL-S/

Neumann, E. K., & Quan, D. (2006). BioDash: A semantic web dashboard for drug development. In *Proceedings of the Pacific Symposium on Biocomputing* (pp. 176-187).

Núñez, S., de Andrés, J., Gayo, J. E., & Ordoñez, P. (2008). A semantic based collaborative system for the interoperability of XBRL accounting information. In *Emerging technologies and information systems for the knowledge society* (LNCS 5288, pp. 593-599).

Pietriga, E., Bizer, C., Karger, D., & Lee, R. (2006). Fresnel: A browser-independent presentation vocabulary for RDF. In *The semantic web - ISWC 2006* (LNCS 4273, pp. 158-171).

Prud'hommeaux, E., & Seaborne, A. (2008). *SPARQL query language for RDF*. Retrieved from http://www.w3.org/TR/rdf-sparql-query

Quan, D., Huynh, D., & Karger, D. (2003). Haystack: A platform for authoring end user semantic web applications. In *The semantic web - ISWC 2003* (LNCS 2870, pp. 738-753).

Richardson, L., & Ruby, S. (2007). *Restful web services*. Cambridge, MA: O'Reilly.

Sabestinal, S. (2009). *Semantic decision support system (DSS) and portal for palm oil industry*. Retrieved from http://www.semanticuniverse.com/articles-semantic-decision-support-system-dss-and-portal-palm-oil-industry.html

Tauberer, J. (2008). *What is RDF and what is it good for?* Retrieved from http://www.rdfabout.com/intro

[1] http://rhizomik.net/rhizomer
[2] Eclipse, http://www.eclipse.org
[3] Paggr - Dashboards for a Web of Data, http://paggr.com/about
[4] Simile Timeline, http://simile.mit.edu/timeline
[5] Electronic Data Gathering, Analysis, and Retrieval system, http://www.sec.gov/edgar.shtml
[6] ReDeFer project, http://rhizomik.net/redefer
[7] BizOntos, http://rhizomik.net/ontologies/bizontos
[8] SemanticXBRL, http://rhizomik.net/semanticxbrl
[9] http://www.freebase.com
[10] http://www.opencalais.com
[11] http://rhizomik.net/semanticxbrl/
[12] http://linkeddata.org

Author Biography

Dr. Roberto García got his MSc. in computer science from the Universitat Politècnica de Catalunya (UPC). His MSc. thesis, completed the year 2000, developed a distributed knowledge management system using Semantic Web technologies. Then, he completed a Master in E-Commerce and collaborated in the first steps of a web and interactive systems company before returning to the academic world. In 2001 he got a research assistant position at the Universitat Pompeu Fabra (UPF) and carried out his research in Semantic Web based media management, which concluded with his PhD thesis in 2006 about a Semantic Web approach to DRM. Now, he is an associate professor at the Universitat de Lleida (UdL) and member of the GRIHO research group, where he is combining his previous research lines with trying to get the Semantic Web in touch with "real-world" end-users. More details and publications at http://rhizomik.net/~roberto