Building a Usable and Accessible Semantic Web Interaction Platform

Roberto García[1], Juan Manuel Gimeno[1], Ferran Perdrix[1,2], Rosa Gil[1], Marta Oliva[1], Juan Miguel López[1], Afra Pascual[1], Montserrat Sendín[1]

[1] *Universitat of Lleida. Jaume II, 69. 25001 Lleida, Spain*
{rgarcia, jmgimeno, ferranp, rgil, oliva, juanmi, apascual, msendin}@diei.udl.cat

[2] *Segre Media Group. Del Riu 6. 25007 Lleida, Spain*
fperdrix@diarisegre.com

Abstract. Semantic Web applications take off is being slower than expected, at least with respect to "real-world" applications and users. One of the main reasons for this lack of adoption is that most Semantic Web user interfaces are still immature from the usability and accessibility points of view. This is due to the novelty of these technologies, but this also motivates the exploration of alternative interaction paradigms, different from the "traditional" Web or Desktop applications ones. Our proposal is realized in the Rhizomer platform, which explores the possibilities of the object-action interaction paradigm at the Web scale. This paradigm is well suited for heterogeneous resource spaces such as those common in the Semantic Web. Resources, described by metadata, correspond to the objects in the paradigm. Semantic web services, which are dynamically associated to these objects, correspond to the actions. The platform is being put into practice in the context of a research project in order to build an open application for media distribution based on Semantic Web technologies. Moreover, its usability and accessibility have been evaluated in this real setting and compared to similar systems.

Keywords. Semantic Web, interaction, usability, accessibility.

1. Introduction

For a complete success of the Semantic Web it is important for it to be adopted by a critical mass of "real world" end users, i.e. users outside the Semantic Web research and development community. Nowadays, this has not happened yet and, as some reports point out [1], this is due in part to the fact that end users find it very difficult to use. Even researches and advanced users of the Semantic Web community find it complicated [2].

The Human Computer Interaction (HCI) discipline proposes a methodology specially focused on this purpose: the User Centred Design (UCD) [3], which is applied with the aim of obtaining usable products. Usability is defined as the degree of effectiveness, efficiency and satisfaction when a product is used by certain users to achieve specific goals within a defined context of use [4].

One of the main reasons why there are so many usability issues in the Semantic Web is because its nature requires changes in the way interaction is sustained, especially due to the fact that it is based on heterogeneous and unanticipated data. Previous systems, even Web-based systems, are commonly based on homogeneous data whose characteristics are known when the interface is being developed.

For instance, traditionally, many interactive systems have been based on the Action-Object [5] paradigm: first, the user selects the action he wants to carry out from pull-down lists that organise the available actions in a hierarchical manner. Then, the user selects the object over which the action should be carried out. For instance, the user first selects the "Open" action from a menu and next the document this action should be applied to.

This is a quite usable interaction model when there is a conceptually homogeneous set of objects to which actions are applied. If this is not the case, it is difficult to maintain a clear arrangement of actions because, firstly, it is difficult to organize it hierarchically and, secondly, because it requires the user to deal simultaneously with a great amount of them in order to find the one he is interested in. This provokes user's cognitive overload and, consequently, usability decays.

However, the Semantic Web promotes and facilitates the creation of very heterogeneous object sets due to the fact that one of its greatest strengths is the ability to integrate multiple sources of data. Consequently, a Semantic Web application that tries to take advantage of this fact will be usually based on a set of heterogeneous objects.

To follow an Action-Object interaction paradigm in these cases will frequently result in a less usable Semantic Web application. By contrast, the alternative based on an Object-Action [6] paradigm is the natural way of interaction in environments characterized by a high degree of heterogeneity of the objects being manipulated.

In this case, the interaction begins when the user selects an object or a set of objects he/she is interested in. Then, the user selects the action that he/she wants to apply on this object, which is chosen from the set of available actions for it. This paradigm simplifies the interaction and can improve usability in heterogeneous contexts like the ones we can find in many Semantic Web applications. Users find it easier to identify and organise objects than actions. In fact, ontologies are mainly

about objects and, in the case of the Semantic Web, web ontologies can be used to attain this.

On the other hand, the group of available actions for an object can be easily determined from the restrictions defined by these ontologies. Consequently, it is possible to exploit the knowledge captured by Semantic Web ontologies in order to give support to the users while they interact under an Object-Action paradigm, freeing them from this burden so they can concentrate on more productive tasks. This approach is especially appropriate in very heterogeneous domains, for instance those resulting from the integration of data coming from different sources.

A platform that puts this approach into practice in the context of the Semantic Web is described in Section 2. There are details about how it faces metadata browsing, metadata edition, the linking between objects and actions, annotation for metadata generation and usability and accessibility issues. Then, a scenario where this platform is been applied is introduced in Section 3, the OMediaDis research project. The accessibility and usability evaluations of these preliminary results are presented in Section 4. Finally, conclusions and future plans are presented in Section 5 and Section 6 respectively.

2. The Rhizomer Platform

Rhizomer[1] is a platform that facilitates building Web applications that help users publish, query, browse, edit and interact with semantic data. Concretely, it gives support to 7 typical tasks of Semantic Web end-users. Here, end-users stand for users with no or limited knowledge about the Semantic Web. Particularly, we don't include domain experts, which might neither have knowledge about the Semantic Web but whose main task is to work with ontologies. The end-user tasks supported by Rhizomer are:

- **Search**: pose a semantic query using HTML forms, which are dynamically generated and user customisable, and obtain resource descriptions rendered as HTML, as shown in Section 2.2.
- **Browse**: navigate through the graph of data retrieving fragments of manageable size and rendering them as interactive HTML. More details are available from Section 2.3.
- **Annotate**: provide new semantic metadata describing a resource, or edit existing one, using HTML forms that assist the user during this process. More details about how users edit metadata are available from Section 2.4 and details about how metadata is generated semi-automatically from Section 2.6.
- **Mashup**: mix two or more pieces of metadata about common resources, or resources similar in some sense, e.g. they all have geographical coordinates or are situated in time and can be placed together in a map or timeline respectively. This simple mashups correspond to two of the interaction services detailed in Section 2.5.

[1] Rhizomer, http://rhizomik.net/rhizomer

- **Share**: upload, update and delete pieces of content (HTML, images, videos, documents, etc.). This is done also through the REST interface and in order to make it more usable an online HTML editor and interactive content uploader is integrated into Rhizomer, concretely FCKEditor[2].
- **Map**: define simple mappings between concepts from different ontologies. This tasks is performed using the same means that for metadata edition but in this case what the user edits is the definition of a class or property, instead of an instance definition.
- **Transact**: generically, this task includes any user action that changes the state of a real-world entity or of a resource in a system outside Rhizomer. Rhizomer features mechanisms that facilitate integrating external web services. More details about the implementation of actions as Semantic Web services are available from Section 2.5. Though these external services are initially considered a transact task, some of them might give support to any other of the tasks previously introduces, apart from being a Transact from the point of view of Rhizomer as a system. As the focus is placed on tasks from the point of view of the user, Transacts should be analysed on a case-by-case basis and characterised as one of the previous tasks if it is possible.

2.1. Technologies and Architecture

From the technological and architectural point of view, the objective is to build a generic web portal, not constrained to a particular application domain or data schema, inspired by Web 2.0 concepts but based on a Semantic Web data model.

In order to obtain a browser based solution, while maintaining a great range of interaction possibilities; AJAX [7] is the client side choice. The server counterpart looks for simplicity and provides a set of really simple services on top of a Resource Description Framework (RDF) metadata store for query, insertion, update and deletion operations implemented through REST [8] commands. The Rhizomer server architecture is shown in **Fig. 1**.

Queries, based on the semantic query language SPARQL [9], are sent to the server using a GET command. However, in order to add support for the other operations, a new range of functionalities have been added to a common SPARQL endpoint: the HTTP PUT and POST commands are used for insertions and updates; the DEL command is used for deletions.

The whole user experience is built on top of these operations. In order to increase usability, RDF is completely hidden. End-users are used to interact through their browsers with HTML web pages. Consequently, Rhizomer incorporates a generic transformation from RDF to HTML, based on an Extensible Stylesheet Language Transformation (XSLT) and detailed in Section 2.2. 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 2.4. Updates, and new metadata generation, are carried out through semantics-enabled HTML forms that also hide the burdens of RDF metadata from users.

[2] http://www.fckeditor.net

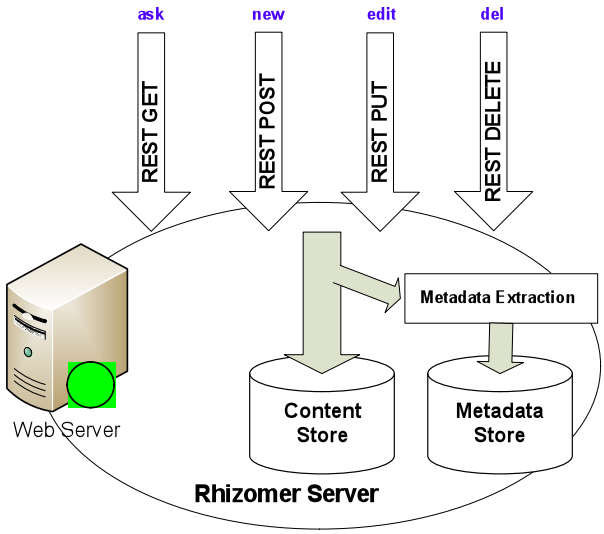


Fig. 1 The Rhizomer Server architecture

The previous metadata management operations and HTML rendering facilities provide a very generic way to deal with the object part of the Object-Action Interaction Paradigm. RDF metadata is the way to describe objects and this metadata is structured using ontologies. Moreover, none of these operations or rendering facilities is specialised in a particular kind of metadata, schema or ontology. This constitutes the object part of the paradigm.

In order to deal with the action part, Rhizomer incorporates Semantic Web services. Each action corresponds to a Semantic Web service that incorporates in its description the constraints an object must satisfy in order to be a valid input for the service. Consequently, the semantic description of the objects, RDF metadata describing them, is considered in order to determine which actions can be applied to them.

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 RDF metadata describing different kinds of objects while being easily extensible in order to incorporate specialised ways to view and interact with particular kinds of them. More details about this mechanism are available from Section 2.5.

2.2. Metadata Search

With Rhizomer, users can perform semantic queries without any knowledge of semantic query languages. All that they need to know is 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. Moreover, users can add other properties that, without being specific, might also apply to

resources of that kind. Each property corresponds to a form input that the user can fill in order to retrieve all resources with that property valued with the input filler.

Query results are those resources satisfying the search criteria. However, result pages show more information than just the identifiers of the retrieved resources. In order to provide more context to the user, each resource is presented together with its description. The user does not face these descriptions as raw data, the user is presented an HTML rendering of the descriptions where all identifiers are substituted by human-readable labels. Moreover, the HTML rendering allows browsing the resources related to the retrieved ones, as detailed in the next subsection.

2.3. Metadata Browsing

Browsing is the basic interaction paradigm in the Web. It is based on the successive visualisation of Web pages following the links that connect them. Pages and links are the main building blocks upon which the interaction is built. However, this browsing paradigm should change because 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 triplets. Moreover, the frontiers among documents are very fuzzy in the Semantic Web: usually, many documents are combined in order to get a coherent graph.

Thus, 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 that keep anonymous resources associated with the identified resources that contextualise them. Following this approach, it is possible to construct fragments for any graph starting from any non-anonymous node. For instance, for the metadata that describes a piece of content, the starting point is the node that represents it and that is the subject for all the triples that describe it. This node has an ID and consequently is not anonymous.

All the triples that start from this node are part of the fragment. Next, all the triples that describe objects that are anonymous are also added to this set. This happens for all nodes that are only identifiable in the context of the starting node. For instance, Fig. 2 shows how an example graph would be fragmented following this approach. As it can be seen, there are two fragments, each one corresponding to one identified resource that is 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. On the contrary to the address, it is shown independently because it is not anonymous.

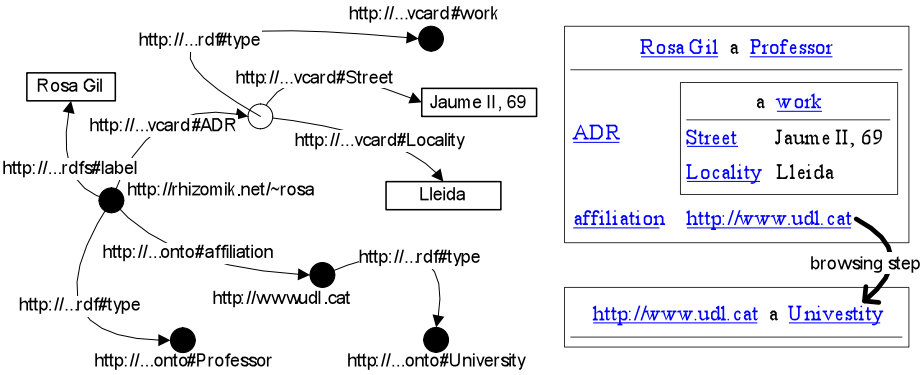


Fig. 2 Fragmentation of an example RDF graph

The resulting fragments are similar to the ones obtained by the Minimum Self Contained Graph (MSG) approach [10]. However, MSGs are not intended for graph browsing but for fragmenting the graph in order to facilitate metadata digital signatures and graph comparison in an incremental way for synchronisation.

The main difference is that MSGs are built from a given triple, not from a resource, so they do not keep all the metadata about a resource in the same fragment. This makes it very difficult to browse a graph using MSGs in a coherent way. Moreover, in order to make the results more usable, Rhizomer fragments include all the labels for the involved resources so, when they are rendered to the user, all the URIs are replaced by labels if they are available.

In order to show fragments to users, they are rendered using HTML that can be viewed using a web browser, a tool users feel comfortable with. In order to generate HTML from RDF, fragments are serialised as RDF/XML that is 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.

The fragmentation makes it possible that the resulting RDF/XML maintains all related triples together, even those for the anonymous resources included in each fragment. Consequently, it is possible to show them like a series of HTML tables, one for each fragment corresponding to the description of an identified resource, that contain nested tables for the descriptions for the anonymous resources contained in the fragment. The RDF to HTML transformation can be tested at the ReDeFer[3] project web site.

Finally, the identified resources and properties, for which just the available labels have been included in the fragment, are shown as HTML links that allow continuing the browsing experience. If the user is interested in any of them, by clicking on them the fragment for the corresponding identified resource is retrieved and rendered as HTML. More details about semantic metadata browsing with Rhizomer are available from [11].

[3] ReDeFer, http://rhizomik.net/redefer

2.4. Editing Metadata

The previous fragment-based approach, besides being the foundation for browsing, allows constraining, to a limited set of triples, the metadata editing and deletion actions. This way, it is possible to implement editing actions as the replacement of a given fragment, the one being browsed when the user clicks the *edit* link, with the one resulting from the editing process. The same applies for the deletion action. In this case, all the triples for the fragment being browsed are removed from the metadata store.

On the other hand, there is also an option that facilitates metadata creation based on a "create from example" approach. It makes possible to create a new description based on the one being browsed. The user should provide a new URI for the resource being described and edit the values generated automatically from the example in order to adjust them to the resource being described.

All these operations (editing, deletion and creation) are also carried out through an HTML interface. In addition to the RDF to HTML transformation, the Rhizomer platform also 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 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 little 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 that are not constrained to a particular resource type plus all the properties constrained to the types of the resource being edited. The future plan is to improve this support in order to assist users during the whole editing process, as it is detailed in future work presented in Section 5.

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. This algorithm implements the reverse transformation, in this case from HTML Forms to the RDF metadata they represent as a result from the previous RDF to HTML Form transformation and the form filling carried out by the user before submitting it.

The algorithm generates the RDF triples for the form. Each form field corresponds to a property, whose URI is captured by the field name, and whose valued is the field filler. All properties refer to the URI of the resource being edited except for anonymous resources that are marked with a hidden form field. This completes the roundtrip for RDF metadata editing from RDF to HTML forms and back to RDF. More details about semantic metadata edition with Rhizomer are available from [11].

2.5. 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 static object part of the Object-Action paradigm. The user can pose queries to access the descriptions of the resources managed by the system and browse through the semantic metadata that describe them.

Once the object (or objects) of interest is located, the actions that the user can do upon it are shown to the user following the Object-Action paradigm. 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.

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 [12] 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.

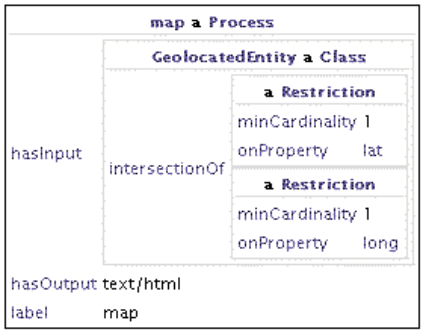
In fact, in the current state of the system, only the class *Process* and the properties *hasInput* and *hasOutput* (defined in OWL-S) are used. *Process* allows identifying the resources that correspond to web services that can be invoked from Rhizomer. Their URIs correspond to the service's access 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 1, 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 1 Description of a geographical information visualization service (Left: Rhizomer rendering. Right: RDF/XML source)




```
<rdf:RDF ...
  xmlns:process="…/services/owl-s/1.1/Process.owl#"
  xmlns:pos="…w3.org/2003/01/geo/wgs84_pos#">
<process:Process
      rdf:about="…/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. In the next section, a specialised web service for the visualisation of multimedia resources is shown in the context of a business application using Rhizomer.

[4] Simile Timeline, http://simile.mit.edu/timeline

2.6. Annotation

As it has been shown, all the interaction is based on the semantic metadata describing the resources managed by the platform. Consequently, when content is uploaded into the platform it is also semantically enriched with annotations. These annotations constitute metadata that describe the content and that is necessary in order to drive user's interaction with the platform.

These enrichment processes, implemented as different plug-ins, can use information that already exists in the platform and external services to generate the metadata, e.g. OpenCalais[5]. For instance when uploading a news article: first, its content can be analysed in order to detect proper names referring to places; then, those places can be searched for in a geolocation service to get their coordinates; finally, the original article can be annotated as referring to places in those coordinates. The reasoner can use these annotations to detect that the article as referring to some *GeolocatedEntities,* allowing for a map-view to be available for them.

These processes are semiautomatic: sometimes, human intervention is needed mainly to disambiguate different interpretations of an entity. For instance, if there exist different places with the same name, the platform presents all the possible referents and asks the user to choose for the right place the article is referring to. In the domain of news websites due to its implicit structure there have been defined metadata extraction algorithms [13] that exploit such structure.

For other types of media such as audio content corresponding to hourly news flashes, it is possible to extract metadata from the text transcript. This transcript can either be automatically generated from the audio or be based on the notes given to the narrator. The former case also requires a manual validation of the transcript and was developed in a previous research project, S5T [14].

Additionally, metadata can already be present in the uploaded content. For instance, without abandoning the geolocation context, some digital cameras include a GPS that can associate to each photograph the coordinates of the place it was taken. If this photograph is uploaded as part of an article, those coordinates can be associated with it.

Some enrichment plug-ins can exploit content already present in the platform, even information provided by the users of the platform. For instance, the same entities that have been detected in the content can be used to find content that refers to the same places, people, etc. Moreover, if some users have tagged those related news, those tags can be suggested as possible ones for the news that is being uploaded. These suggestions contribute to the network effect of social tagging making it easier to evolve a common folksonomy. In order to reduce the number of synonymous tags, some measure of semantic similarity will be needed [15]. More detail about annotation are available from [16].

[5] Thomson Reuters Calais service, http://www.opencalais.com

2.7. Usability and Accessibility

The Human Computer Interaction (HCI) discipline proposes a methodology specially focused on the usability issue: the User Centred Design (UCD) [3]. According to this discipline, user needs are taken into account from the beginning and throughout the whole development process, with the aim of obtaining usable products. In order to put UCD into practice, different prototypes and models are constantly developed and evaluated, both by usability experts and final users, in terms of guaranteeing usability.

Usability evaluation is a major aspect in any UCD methodology. So, it is clear that to ensure that the Rhizomer platform is usable is essential to assess its level of usability. In the literature, it is possible to find several usability evaluation techniques and among them there is the heuristic evaluation, which has been applied in order to evaluate the Rhizomer usability.

Regarding accessibility, it is remarkable its demographic importance. For instance, according to Eurostat [17], from a total population of 362 million people in Europe in 1996, a 14,8% of the population between 6 and 64 years old had physical, psychological or sensorial disabilities. Also, there are also powerful legal reasons in order to develop accessible web user interfaces. In any case, just for ethical and moral reasons, it is important for a web platform such as Rhizomer to take accessibility into account.

Apart from overall web content accessibility, the content generated by the "action" plug-ins included in the platform (such as map or timeline) should also be analysed. Although they are included in the platform, these plug-ins are independent components that are reused in the platform. The accessibility of uploaded contents is also a factor to be taken into account, which is for instance addressed in the case of the semantically annotated transcript for audio content. Finally, there is also concern about the accessibility of the AJAX technology used in the platform, as existing accessibility guidelines [18] establish that web pages should be usable for users with disabilities without the necessity of having Javascript activated in their web browsers.

A usability evaluation and a deep analysis of the accessibility of the platform have been carried out. A detailed description of the methods used and the results are presented in the Evaluation Section.

2.8. Related Work

There are very few tools that provide the range of functionalities and interaction services presented in this section in a flexible way. One of the tools that provide similar functionality is the extensible Semantic Web browser Haystack [19]. However, this is not a Web application; it is a desktop application build on top of the Eclipse[6] framework.

A similar but Web-based solution that has been recently announced is Paggr[7]. It is a framework for semantic dashboards development based on Semantic Web technologies and PHP. However, Paggr is still under development so it has not been

[6] Eclipse, http://www.eclipse.org

[7] Paggr - Dashboards for a Web of Data, http://paggr.com/about

possible to evaluate it with detail. In any case, it seems to concentrate on semantic web browsing and querying and currently does not feature other interaction services.

Other related tools are ODESeW [20], a Semantic Web application development platform, or semantic wikis like the semantic extension for Media Wiki [21], which mix wiki mark-up and semantic annotations. Even more specific in functionality are RDF Browsers like Tabulator [22] or Disco [23], which just provide browsing capabilities and in some cases metadata edition.

Moreover, many Semantic Web browsers show all the triples from a Semantic Web document at once, in the case of Tabulator 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 [24]. However, our objective is a simpler and more polyvalent browsing mechanism that, though it might lack the guidance provided by facets, it can deal better with heterogeneous information spaces.

Moreover, faceted and other Semantic Web browser tend to make it difficult to navigate through metadata structures that feature many anonymous resources, as it is the case for the semantic metadata managed in the OMediaDis project that is described in Section 3. Usually, they show anonymous resources and their associated metadata in isolation with the consequent loose of context for the user. This is due to the fact that anonymous resources commonly lack labels or other clues that help identifying them.

From the point of view of the additional interaction services beyond search and browse, there are some tools that already provide these specialised views on different kinds of semantically described resources, such as Tabulator [22] or Exhibit [25]. However, the range of alternative views is fixed a priori and new views are incorporated in an ad-hoc way to the underlying RDF metadata browsing facilities.

In the Rhizomer case, this flexibility is attained through semantic descriptions of the available interaction services based on OWL-S. However, different semantic web services platforms have been evaluated, mainly WSMO [26] and SAWSDL [27]. All of them are too complex for the simple requirements of the platform.

The complexity does not lie in the semantic model that these platforms provide, but in the fact that all of them are based on web services standards such as WSDL/SOAP [28]. This kind of semantic web services is more appropriate in business environments but it is over-complex for the Rhizomer platform in which the actions will mainly be used to implement data visualization services.

Besides, many of the publicly available web services, e.g. Google Maps, are not available as WSDL/SOAP. In fact, it seems that services based on WSDL/SOAP are being displaced by REST ones [29]. For instance, big web services providers (like Google, eBay or Yahoo!) are basing their services on REST and export them by APIs in JavaScript or other languages. This approach is appropriate when strong requirements of security do not exist and a simple development model is an objective. In any case, it is also possible to implement security mechanisms over REST [8].

3. OMediaDis Application Scenario

OMediaDis [30] is a research project whose aim is to build an open platform for content distribution management. On the one hand, it is intended for small content providers and professionals. For them it provides services like content publishing, semantic indexing, assisted metadata edition, copyright management and use monitoring. Most of this services are provided by the Rhizomer platform, which is complemented with a semantic copyright management module [31] and a watermarking service in order to provide the two last services.
On the other hand, the platform is also intended for a full range of content consumers and distributors, which might be individual users but also small to medium media groups like Segre[8], which participates in the project. For them, the services are content search and navigation, annotation, recommendation and content negotiation. All these services are based on Rhizomer except for the last one, which is provided by the copyright management module.

This paper does not get into detail about those services that are not provided by Rhizomer. In the next subsections, some of the Rhizomer functionalities in the context of the OMediaDis project are presented. First of all, it is shown how content metadata can be browsed. Then, there is an action specific for this project that allows interaction with the semantically annotated transcript of audio content. This action is dynamically associated to all audio objects that have a transcript. Finally, it also shown how the same generic interface can be used in order to browse additional metadata and ontologies that capture the application domain knowledge.

3.1. Content Search and Browsing

The users can start by building a query for the particular kind of object they are interested in, e.g. audio. They do not need to know the semantic query language syntax, the underlying ontologies are used in order to build an HTML form. In order to do that, a similar process to that followed for metadata edition is carried out, c.f. Section 2.4. Initially, the form has one input field for each of the properties specific for that kind of object or its superclasses, e.g. *transcript*, as defined in the underlying ontologies.

Moreover, the user can interactively add other properties that also apply to that kind of object, but are not specific to them, for instance *title*. Each added property corresponds to a new input field in the form. The user fills the input fields in order to constraint the values for the corresponding properties, which implicitly defines the query that will be generated automatically when the user submits the form.

Once a query is executed, metadata associated with the selected resources is shown by means of the HTML interface for metadata browsing, as it is shown in the left part of Fig. 3. In the case of the OMediaDis project, multimedia metadata is based on the Dublin Core[9] for editorial metadata, i.e. title, date, author, etc. and on an ontology for

[8] Diari Segre Media Group, http://www.diarisegre.com
[9] Dublin Core, http://dublincore.org

the standard IPTC News Subjects[10] for genres. For content-based metadata, particularly content decomposition based on audio transcripts, a MPEG-7 Ontology is used [32].

All the resources and properties that appear in the metadata HTML view are links that allow the user to retrieve additional metadata about the clicked resource. For instance, the contents described in Fig. 3 refer to the "agriculture" genre. If the corresponding link is followed, the metadata for the corresponding resource is retrieved from the IPTC news topics ontology and shown. Consequently, it is possible to browse the descriptions for the news items managed by the OMediaDis application and the descriptions for the terms used in these descriptions.

3.2. Transcript-based Interaction Service

In addition to the metadata browsing facility, which provides a way to interact with the objects by means of the object-action paradigm, there are some web services, such as the ones described in Section 2.5, that provide some customized actions. Additionally, there is a specific action for the OMediaDis scenario that is enabled for audiovisual resources, i.e. resource of type *mpeg7:AudioType*, with an associated *transcript* property. The corresponding web service provides a view, shown in the right part of Fig. 3, which allows additional interaction possibilities through the transcript semantic annotations automatically generated [14]. Moreover, this view also improves the accessibility of the managed content.

Fig. 3 Metadata view (left) and transcript view (right) available through the "play" service

This view allows rendering audio and video content and interacting with it through a clickable version of the audio transcript. Two kinds of interactions are possible from the transcript. First, it is possible to click on any word in the transcript that has been

[10] Semantic Newspaper, http://rhizomik.net/semanticnewspaper

indexed in order to perform a keyword-based query for all the pieces of content whose transcript contains that keyword.

Second, the transcript is enriched with links to the ontology used for semantic annotation. Each word in the transcript whose meaning is represented by an ontology concept is linked to a description of that concept. Then, that description is presented as it is detailed in the next subsection.

For instance, the transcript includes the name of a place that has been indexed and modeled in the ontology. Consequently, it can be clicked in order to get all the audiovisual items where that place appears or, alternatively, to browse all the knowledge about that place encoded in the corresponding domain ontologies.

3.3. Domain Knowledge Browsing

When the user chooses to browse concepts in the annotated transcript, the interaction gets back to the generic metadata browsing view. Then, the user can browse the ontologies used to annotate the transcripts. Each browsing step gets the user through these ontologies.

Consequently, continuing with the example in the previous subsection, when the user looks for the available knowledge about that place, an interactive view of the RDF data about it is shown. Currently, we are using concepts from DBPedia [33] in order to semantically annotate content whenever possible.

This way, the user can benefit from the modelling effort already made in Wikipedia and formalised into DBPedia and, for instance, be aware of the coordinates of the place in order to situate it into a map or the regions that place belongs to. The subsequent browsing steps, e.g. following the links to the containing regions or related places, will show additional domain knowledge from the annotation ontologies, in this case DBPedia.

In addition to this interactive navigation of all the domain knowledge, at any browsing step, it is also possible to get all the content annotated using the concept currently being browsed. This action might bring the user back to the transcript-based view. Thanks to this dual browsing experience, the user can navigate through audiovisual content and the underlying domain knowledge in a complementary an interwoven way.

4. Evaluation

In order to assess the usability and accessibility of the Rhizomer platform and of its application in the context of the OMediaDis project, an accessibility and an usability evaluation have been conducted. Both are detailed in the next subsections.

4.1. Accessibility Evaluation

Although different web accessibility evaluation methodologies exist [34], the most accepted one is the one provided by the W3C [35]. Moreover, accessibility evaluation

has been traditionally based on revising the fulfilment of the Web Content Accessibility Guidelines (WCAG) proposed by the W3C [18]. Thereby, this accessibility evaluation methodology has been used to evaluate the accessibility of the Rhizomer platform. It must be noted that the accessibility evaluation was performed in late February 2009, so future changes performed on the Rhizomer platform should have influence on its accessibility. This W3C methodology defines a series of steps in order to evaluate web accessibility.

4.1.1. Determine the scope of the evaluation

Web pages from the core Rhizomer platform and related to different elements such as maps and multimedia were selected as a representative sample of pages in the platform. All analyzed web pages were in the public part of the platform, so they can be freely accessed by any user.

4.1.2. Use web accessibility evaluation tools

The correctness of (X)HTML content and CSS style sheet standards linked to each web page were analysed using the validation services provided by the W3C. Then automatic accessibility evaluation tools [36] were used, which show a revision of automatically detectable accessibility problems. According to the W3C [18], it is recommended to use at least two different automatic accessibility evaluation tools. This recommendation is based on the fact that, being the accessibility guidelines expressed using natural language, the results provided by different automatic evaluators can differ. In this particular case, TAW[11], Evalaccess[12] and TotalValidator[13] were used as automatic accessibility evaluation tools.

Results for (X)HTML evaluation show that web pages on the platform are correct. However, CSS evaluation indicates that there are several issues regarding the use of no standard elements in the style sheets. This situation is mainly due to the use of external components such as the GoogleMaps API, SIMILE Timelines and Yahoo! User Interface components.

4.1.3. Manually evaluate representative page sample

Automatic accessibility evaluation tools also point out several possible problems that cannot be automatically revised and require manual revision by accessibility evaluators. In this sense, the accessibility checklist was applied on every web page. Use of assistive technology such as screen readers or text browsers is also advocated.

Lynx as a text browser and JAWS as a voice browser were used with this aim. Different existing web browsers were used to check if the web pages were correctly visualized. During the manual evaluation, no major accessibility problems have been found in the core Rhizomer web pages, thus automatic accessibility evaluation results can be considered as a good indicator for overall core Rhizomer platform accessibility.

---

[11] TAW, http://www.tawdis.net/taw3/cms/en
[12] Evakaccess 2, http://sipt07.si.ehu.es/evalaccess2
[13] TotalValidator, http://www.totalvalidator.com

Table 3 shows the different accessibility errors found for the representative sample of core Rhizomer platform web pages. The first column displays the core Rhizomer web pages analysed. Next three are related to the WCAG errors found. Each of these columns represents the amount of errors found for the different levels of priority defined in the WCAG, based on the checkpoint's impact of accessibility. Priority 1 checkpoints represent the checkpoints that a Web content developer must satisfy, priority 2 the ones that should be satisfied and priority 3 are the ones that a web developer may address to enhance accessibility.

Table 3. Errors found during the automatic analysis of a representative sample of the core Rhizomer platform web pages

Web Page	Priority 1	Priority 2	Priority 3
http://rhizomik.net/	0	0	0
http://rhizomik.net/login/login.jsp	0	5	0
http://rhizomik.net/copyright	3	4	0
http://rhizomik.net/?edit	0	0	0
http://rhizomik.net/rhizome	0	0	0
http://rhizomik.net/rhizomer	0	4	0
http://rhizomik.net/copyright?edit	3	3	0
http://rhizomik.net/rhizome?edit	0	0	0
http://rhizomik.net/rhizomer?edit	0	4	0
http://rhizomik.net/s5t	0	5	5
http://rhizomik.net/s5t/login/login.jsp	0	5	0
http://rhizomik.net/s5t/copyright	0	2	1
http://rhizomik.net/s5t/?edit	0	4	5

There are other considerations to be taken into account, such as the accessibility of documents uploaded by users. In this case, no accessibility evaluation has been performed on the documents currently on the platform, mainly PDF format documents, as it would be out of the scope of evaluating the accessibility of the platform itself.

Regarding the accessibility of multimedia elements, it is remarkable the fact that multimedia elements are integrated in the platform in a way that makes them easy to be located. **Fig. 4** shows how multimedia content can be accessed using a text browser. Anyhow, options to stop or pause the audio content cannot by performed by text browsers as they are performed using a Flash components. In this case, these options would only be available for a user with disabilities by downloading the file and playing it in his/her own media player.

There is also external content which is not accessible using textual browser and that can be considered as a handicap for people with disabilities. This situation is due to the presence external components such as information from the DBPedia, SIMILE Timetables and Google maps. These components are integrated by means of Javascript based libraries so accessibility compliance fails in all web pages that make use of them. WCAG 6.3 checkpoint establishes that web pages must be usable if programmable objects such as scripts are turned off or not supported. It is remarkable that transcripts for audio content do not show any lack of accessibility when accessed

by text or voice browsers as they are linked from a HTML page without any Javascript.

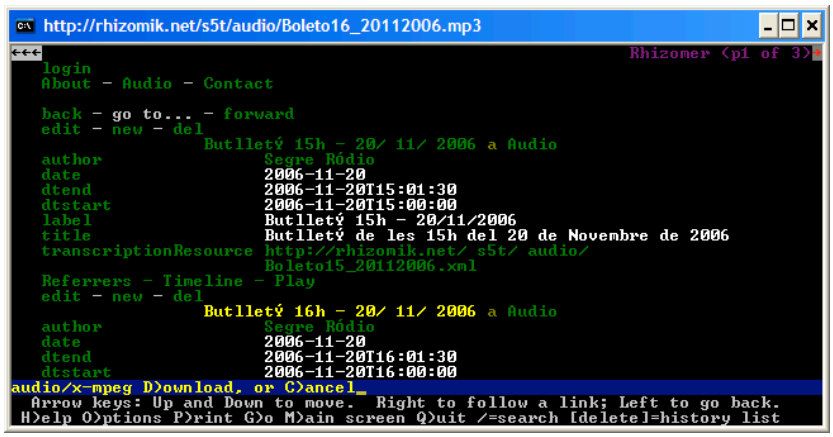


Fig. 4. Downloadable audio content in Rhizomer using Lynx text browser

4.1.4. Comparison with other semantic web environments

This subsection is aimed at providing a comparison among accessibility results from different semantic web platforms. The platforms considered for this study have been DBPedia[14], Multimedian[15] and Mspace[16]. The last version of Tabulator[17] was also intended to be analysed, but its Javascript based navigation has made the crawling process impossible to apply, as the navigation in Tabulator did not point out to different web pages that could be crawled. It must be noted that this kind of navigation implies an accessibility problem by itself, as Tabulator can not be navigated using specialized navigators such as textual browsers.

Rhizomer was also included in the analysis in order to be compared with the other platforms. Anyway, it must be taken into account that results provided in previous subsections are more reliable for Rhizomer, as they also include manual accessibility evaluation.

In order to provide a valid comparison, a preliminary review of accessibility was performed according to the methodology provided by the W3C [37]. Due to the relatively small number of web pages on each platform analysed, all selected platforms were crawled to extract the web pages that constituted each platform. A limit of 250 web pages was established to perform the crawling in the unique case where analysed web sites was larger than established limits, in this case the DBPedia.

All crawled web pages constituted the sample of representative web pages. These web pages were then examined with graphical and specialized browsers. Finally, two

[14] http://dbpedia.org/
[15] http://e-culture.multimedian.nl/demo/session/search
[16] http://demo.mspace.fm/
[17] http://dig.csail.mit.edu/2005/ajar/release/tabulator/0.8/tab.html

automatic accessibility evaluation tools were used to check the accessibility of each web page, in this case TAW and TotalValidator.

It must be noted that even if no manual accessibility evaluation is performed, the preliminary review of accessibility provides a landscape about the state. In this sense, references can be found in the literature about accessibility measurement metrics based on the WCAG that show small error rates when compared with manual accessibility evaluations [38, 39, 40].

Table 4 summarizes the results from the performed analysis. In this case, the mean and variance of the different priorities of accessibility errors was calculated for each web site. The number of analysed pages for each platform has also been included. It must be noted that changes performed on all platforms since the evaluation was performed may have had influence on the accessibility of them.

Table 4. Automatic accessibility evaluation results for the different analysed platforms

Platform	Mean prior. 1	Variance prior. 1	Mean prior. 2	Variance prior. 2	Mean prior. 3	Variance prior. 3	#pages
Rhizomer	0,180	0,013	7,500	0,190	1,060	0,037	50
DBPedia	0,016	0,001	4,020	0,001	0,996	0,000	250
Multimedian	1,400	0,072	9,636	0,175	2,309	0,023	55
Mspace	6,286	0,270	11,750	0,332	1,821	0,155	28

Results of performed analysis show DBPedia as the platform with the best accessibility results overall. It shows the best results for all priority accessibility errors. Furthermore, it also shows the lowest variance for all the different priorities. It shows that most analysed pages share a similar number of accessibility errors. In this sense, solving most common accessibility errors found for a single web page could result in an improvement for the whole website, as web pages throughout the website share similar accessibility errors.

Anyhow, it must be pointed out that, as a semantic version of the Wikipedia, DBPedia shows little web pages where multimedia content such as videos is handled and shows little use of Javascript and AJAX. In this sense, the structure and requirements for the website makes it less prone to have accessibility errors compared to the rest of analysed websites. In the case of DBPedia, accessibility errors found relied on the use of deprecated and the lack of summary in tables.

Rhizomer is the second with best accessibility overall. Results in this evaluation are quite similar to the ones shown in previous subsection. In this sense, most accessibility errors found relied on the use of external data by means of Javascript. Low values shown in the variance for the three kinds of accessibility errors imply that errors found are quite similar for all the web pages in the platform. It can be noted that, regarding Rhizomer platform, there are differences between current accessibility analysis and the one depicted on table 3 from previous subsection. This fact can be considered as normal, as web pages out of the scope of a representative sample can influence the accessibility of the overall website.

Regarding Multimedian, accessibility errors found increase notably regarding previously mentioned platforms. The fact that it includes multimedia content managed by means of Javascript has supposed notable accessibility problems. Main accessibility problems found include ensuring that equivalents of dynamic content are

updated and available as often as the dynamic content, providing text equivalents for non-text elements and using absolute units instead of relative ones in markup language attribute and style sheet property values.

Finally, Mspace shows the most accessibility errors per page among all compared platforms. Main problems found are related with ensuring that web pages are usable when scripts are turned off, providing text equivalents for every non-text element and avoiding deprecated attributes of W3C technologies.

4.2. Usability Evaluation

In order to make the system easier for users to reach platform functionality (final consumers as much as other actors in the value chain), a heuristic evaluation was carried out to detect possible usability problems. In the rest of the section, we present the methodology that has been applied, the procedure that has been followed and the results that have been obtained.

4.2.1. Description of the method

Usability evaluation is a major aspect in any UCD methodology. So, it is clear that to ensure that the Rhizomer platform is usable is essential to assess their level of usability. In the literature is possible to find several usability evaluation techniques, among them there is the heuristic evaluation.

Jakob Nielsen [41] developed heuristic evaluation on the basis of several years of experience in teaching and consulting about usability engineering. It involves the participation of a small set of evaluators, which have to examine the interface in order to judge its compliance with recognized usability principles (the known heuristic principles, or simply "heuristics") following a process of inspection of the user interface. Consequently, this method belongs to the category of usability evaluation methods known as inspection methods.

As afore pointed out, this method does not require recruiting users, which can be burdensome due to the need for arranging an appointment, a place to test them and a payment for their time. On the contrary, it only requires a small set of evaluators (between three and five experts), reducing the complexity, the expended time and thus the cost for evaluation. Furthermore, there is neither especial software nor equipment required. This is why it is popularly considered as a *discount usability engineering* method.

The time required varies with the size of the artifact and its complexity. However, we have to take into account that this method does not evaluate a real use of the system, and thus it does not provide the same feedback than an evaluation with users. This is the reason why it is recommended to combine this method with some user testing. In particular, using heuristic evaluation prior to user testing will reduce the number and severity of design errors discovered by users. Another problem related with this method is that the results are liable to certain subjectivity by the evaluators. This inconvenience can be partially overcome involving a greater number of evaluators.

Today, in the context of usability evaluation of interactive systems, the heuristic evaluation technique is very popular. Independent research [42] has confirmed that

heuristic evaluation is a very efficient usability engineering method. Jeffries et al. found that heuristic reviews identified more usability issues than the other methods used in their study.

The output from using the heuristic evaluation method consists of: (1) a list of usability problems in the interface, particularly those that are difficult to detect by users, so that they can be attended to as part of an iterative design process [43]. This list should include references to those usability principles to which they violate; (2) some suggestions and recommendations about their possible solution.

4.2.2. Followed Procedure

In this case, four evaluators were recruited, which were in charge of examining and judging the interface. Initially a first contact was made with the portal to become familiar with the interface and then the evaluation and rating was conducted based on the heuristics specifically chosen for the portal to evaluate.

In particular, the heuristics adopted are fourteen rules that extend the Nielsen and Molish's ten heuristic rules [44]: H1. Clarity of objectives; H2. Visibility of system status; H3. Match between system and the real world/logic of information; H4. User control and freedom; H5. Consistency and standards; H6. Error prevention; H7. Recognition rather than recall; H8. Flexibility and efficiency of use; H9. Aesthetic and minimalist design; H10. Help and documentation; H11. Search; H12. News; H13. Various; H14. Architecture information.

These heuristics were divided into subheuristics (a set of questions intended to facilitate the exploration of the main heuristic), in such a way that it is possible to form related concept groups and subgroups. A detailed description of each heuristic and subheuristics can be found in [45]. Apart from that, to determine the severity level, two parameters were used: *Impact* and *Frequency*. I*mpact* of the problem was used to estimate in which level users are affected when the problem happens. Frequency shows the frequency with which problems occur. Each parameter can be scored on a scale that ranges from 0 (not a usability problem) to 4 (catastrophe: it is compulsory to fix it) [46]. To facilitate the data collection a template adjusted to these parameters and heuristics was specifically made and delivered to each evaluator. Comments from evaluators were also collected.

The method was performed by having each evaluator inspect the interface alone. This individual revision is accompanied with annotations and punctuations about the severity of each problem detected, in the terms aforementioned. Only after all evaluations were completed individually it was allowed to communicate the different opinions. For that, a meeting was arranged so that the evaluators could analyze the collected data, put together their findings and discuss their points of view, with the aim of drawing some final conclusions. This procedure is important in order to ensure independent and unbiased evaluations from each evaluator.

Conclusions and some recommendations related to the usability problems detected were gathered in a report, which have been ordered and categorized according to their severity level and priority for their solution. The most important ones are presented in the next section.

4.2.3. Results obtained

As a consequence of the review of the qualitative results, evaluators made some improvement proposals:

- *Differentiating types of links*: browsing the Rhizomer platform is based on several types of links. It is used a kind of link for browsing among contents and another for browsing metadata. Although the standard link representation is used in the platform, other types of representation are needed to distinguish which is the kind of destination, and also to identify which part of text is a link. Currently, content links are underlined and metadata links are not. This problem does not fulfil the heuristics H2 (Visibility of system status) and H5 (Consistency and standards).
- *Highlight access to the services offered*: the actions that the user can do upon an object, which are showed to the user following the Object-Action paradigm, must to be highlighted in order to not pass unnoticed for the user. This could be achieved using small icons showing appropriate metaphors or other methods. The heuristics violated are the H2 (Visibility of system status), H4 (User control and freedom) and H7 (Recognition rather than recall).
- *Lack of feedback about current user location in the site*: the Rhizome platform interface is divided into two main areas. The left one is the metadata browsing area and the central-right one is the place where content is shown. In many cases, like when browsing metadata, changes are limited to the right area, less prominent than the central part, so they might get unnoticed by the user. This might cause user disorientation. Currently, this issue has been minimised by loading into the central part, whenever the user browses the available resources of a particular type, the query form dynamically generated for that type, c.f. Section 3.1. In any case, in order to improve user feedback, we are currently considering implementing some sort of breadcrumbs [47] from the user metadata graph traversal. This problem is related with the heuristic H2 (Visibility of system status).

5. Conclusions

The Rhizomer platform provides an interaction environment based on the object-action paradigm that is better suited for heterogeneous information spaces than the traditional action-object paradigm. The platform is based on Web 2.0 technologies on top of Semantic Web metadata and ontologies.

The platform offers a generic RDF to HTML transformation that makes it possible to navigate through semantic metadata and the associated ontologies. Resources and their descriptions constitute the object part of the paradigm while the actions that the users can carry out on these resources are implemented by means of a Semantic Web services based on a REST approach. Actions are associated to objects in a completely dynamic way computed by a Semantic Web reasoner on the basis of the semantic descriptions of resources and services.

This platform is being applied in the context of the OMediaDis research project in order to develop its user interface and many of its services. In this scenario, besides semantic search, metadata browsing and some generic actions such as showing geolocated entities in a map, there is a specialised action for audiovisual content with

a transcript for the audio voice. This service allows to reproduce the content and to see the transcript enriched with semantic annotations for keywords. The annotations can be used in order to retrieve other pieces of content featuring the same keyword or in order to browse the semantic annotations metadata.

From the point of view of the usability of the Rhizomer platform, and of the web applications based on it, it has been possible to take into account the results of the usability evaluation carried out. The first change has been to clarify the explanation of the purpose of the sites as it appears in the home page. The home page has been also improved by the addition of a news mechanism, as also recommended.

Other important changes implemented in the last version of Rhizomer are that now visited links are marked with a different colour and that there is a distinction between links to HTML content (underlined) and links to metadata queries (not underlined but with the link colour and constrained to a very specific region of the page). Finally, the links to the services available for each resource have been highlighted by using a different colour than for the rest of the links. In order to guarantee they are noticed the link colour chosen is the complementary colour to the one chosen for the rest of the links.

Regarding accessibility, it is clear that it is a key aspect to be taken into account for providing support for a human-centred approach in Rhizomer platform. In this sense, the use of semantic data and data integration can be a good chance to improve accessibility. Results for performed accessibility evaluation show that the use of semantic data can be helpful for accessibility, as core Rhizomer web pages show little accessibility problems.

Moreover, the use of automatic transcripts for media content as an external source provides a significant accessibility betterment as the transcription of the media is made automatically without any effort for media content uploaders. Anyhow, external data integration in Rhizomer platform also shows significant accessibility related problems. These problems are hard to be resolved as they are caused mainly by the APIs of such external data that external information providers supply.

Currently, they make necessary the inclusion of not accessible web code in Rhizomer so the external data can be integrated in the platform. Although data integration is always a desirable target for providing better services for users, there is a clear risk of leaving an important amount of users out of it by the lack of accessibility of available technologies.

The cause for the lack of accessibility in the APIs for integrating external data can be located in the difficulties that making web 2.0 technologies (such as AJAX) accessible presumes. In this sense, the recent approval of a second version for WCAG [48] is expected to be helpful as it provides a mean to validate several web 2.0 related accessibility issues. Moreover, W3C is unrolling a new recommendation to develop accessible rich internet applications [49] that is expected to solve some of the difficulties associated with the data integration technologies used in this work.

6. Future Work

The future work focuses on metadata edition features and user testing. In addition to the current assistance when a user tries to add a new property to the current description, the idea is also to assist users when they add property values. Properties ranges and restrictions on them that apply to the kind of resource being edited will be considered in order to propose resources that constitute a proper value for the property.

Moreover, up to now we have focused the use of the annotations to offer different views of the data. In the OMediaDis project we plan to use these metadata to offer recommendation services of content to the users of the platform. These metadata can be used to better characterise both the profiles of the users and the description of the content allowing for a better recommendation system.

The other main objective is to continue with usability and accessibility testing in order to quantify the usability improvements that this approach can produce. Currently, we have the first usability results coming from the heuristic evaluations carried out by usability experts. They have pointed out many usability issues and some improvements have been already implemented as described in Section 5.

However, there are still some remaining issues. Some of them are simple and will require just some development effort. For instance, to integrate a search box in the home page and also the inclusion of dynamic search forms generated from the kind of resource the user is interested in. These automatically generated forms are currently work in progress and benefit from the underlying ontologies and the mechanisms for assisted metadata edition currently available.

There are other issues that require further research and development, which are related to the fact that Rhizomer is based on Semantic Web technologies and this introduces a more complex information architecture. Consequently, we should explore these issues with more detail, for instance in order to determine if it is possible to adapt "classical" information architecture component like breadcrumbs to the Semantic Web. The current work line is that they might be generated from the navigation history that Rhizomer keeps track of in order to provide support to back and forward browser buttons.

Acknowledgements

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

1. Shadbolt, N., Hall, W. and Berners-Lee, T.: "The Semantic Web revisited". Intelligent Systems, Vol. 21, No. 3, pp. 96-101, 2006
2. Heath, T., Domingue J. and Shabajee P.: User interaction and uptake challenges to successfully deploying Semantic Web technologies". In Proc. 3rd International Semantic Web User Interaction Workshop, Athens, Georgia, USA, 2006
3. Granollers, T.: "User Centred Design Process Model: Integration of Usability Engineering and Software Engineering". Doctoral Consortium, INTERACT'03, Zurich, 2003
4. ISO 9241-11. (1998). ISO 9241-11. Ergonomic Req. Part 11: Guidance on Usability. ISO 9241-11
5. Bruner, J.: "Action, Thought and Language". Alliance Psychology, 1989
6. Raskin, J. : "The Human Interface". Addison Wesley, 2000
7. Crane, D., Pascarello, E., James, D.: "Ajax in Action". Manning Publications, 2005
8. Richardson, L. and Ruby, S.: "Restful Web Services". O'Reilly, 2007
9. Prud'hommeaux, E., Seaborne, A.: "SPARQL Query Language for RDF". W3C Recommendation, World Wide Web Consortium, 2008. Available from http://www.w3.org/TR/rdf-sparql-query
10. Tummarello, G., Morbidoni, C., Puliti, P., Piazza, F. : "Signing individual fragments of an RDF graph". Proceedings of the WWW 2005 Conference, pp. 1020 - 1021, 2005
11. García, R., Gimeno, J.M., Perdrix, F., Gil, R., Oliva, M.: "A Platform for Object-Action Semantic Web Interaction". In: 16th International Conference on Knowledge Engineering and Knowledge Management Knowledge Patterns, EKAW'08. Lecture Notes in Computer Science, Vol. 5268, pp. 404-418, Springer, 2008
12. Martin, D. (ed.) OWL-S: "Semantic Markup for Web Services". W3C Member Submission, 2004. http://www.w3.org/Submission/OWL-S
13. Mukherjee, S., Ramakrishnan, I.V.: "Automated Semantic Analysis of Schematic Data". World Wide Web, Vol. 11, No. 4, pp. 427-464, 2008
14. Tejedor, J., García, R., Fernández, M., López, F., Perdrix, F., Macías, J.A., Gil, R., Oliva, M., Moya, D., Colás, J., Castells, P.: "Ontology-Based Retrieval of Human Speech". In: Proc. of the 6th International Workshop on Web Semantics, WebS'07. IEEE Computer Society Press, 2007
15. Maguitman, A.G., Menczer, F., Erdinc, F., Roinestad, H., Vespignani, A.: "Algorithmic Computation and Approximation of Semantic Similarity". World Wide Web, Vol. 9, No. 4, pp. 431-456, 2006
16. García, R., Gimeno, J. M., Perdrix, F., Gil, R., Oliva, M.: "The Rhizomer Semantic Content Management System". In : 1st World Summit on the Knowledge Society (WSKS 2008). Lecture Notes in Artificial Intelligence, Vol. 5288, pp. 385-394, Springer, 2008
17. Eurostat: "Health Stadistics". Office for Oficial Publications of the European Communities, 2002. Retrieved from: http://epp.eurostat.ec.europa.eu/cache/ITY_OFFPUB/KS-08-02-002/EN/KS-08-02-002-EN.PDF
18. World Wide Web Consortium (W3C): "Web Content Acessibility Guidelines 1.0", 1999. Retrieved on June 2009 from: http://www.w3.org/TR/WCAG10/
19. Quan, D., Huynh, D., Karger, D.: "Haystack: A Platform for Authoring End User Semantic Web Applications". In : The SemanticWeb - ISWC 2003. Lecture Notices in Computer Science, Vol. 2870, pp. 738-753, Springer, 2003
20. Corcho, O., López-Cima, A., Gómez-Pérez, A.: "The ODESeW 2.0 semantic web application framework". In : Proceedings of the 15th International Conference on World Wide Web, WWW '06, pp. 1049-1050, ACM Press, 2006
21. Krötzsch, M., Vrandečić, D., Völkel, M.: "Semantic MediaWiki". In : Proceedings of the Int. Semantic Web Conference, ISWC'06. LNCS vol. 4273, pp. 935-942, Springer (2006)
22. Berners-Lee et. al: "Exploring and Analyzing linked dates on the Semantic Web". Proc. of the 3rd International Semantic Web User Interaction Workshop, 2006

23. Bojars, U., Passant, A., Giasson, F., Breslin, J. G.: An Architecture to Discover and Query Decentralized RDF Data. In : Proceedings of the ESWC'07 Workshop on Scripting for the Semantic Web, SFSW 2007. CEUR Workshop Proceedings, vol. 248 (2007)
24. Hildebrand, M., Ossenbruggen, J., Hardman, L.: "/facet: A Browser for Heterogeneous Semantic Web Repositories". In Proc. of the International Semantic Web Conference 2006. Lecture Notices in Computer Science, Vol. 4273, pp. 272-285, Springer 2006
25. Huynh, D. F., Karger, D. R., & Miller, R. C. "Exhibit: lightweight structured data publishing". In : Proceedings of the 16th international conference on World Wide Web, pp. 737-746, Banff, Alberta, Canada, ACM, 2007
26. Roman, D., Keller, O., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., et al.: "Web Service Modeling Ontology". Applied Ontology, Vol. 1, No. 1, pp. 77-106, 2005
27. Farrell, J. and Lausen, H. (eds.): "Semantic Annotations for WSDL and XML Schema". W3C Working Draft, 2007. http://www.w3.org/TR/sawsdl
28. Weerawarana, S., Curbera, F., Leymann, F., Storey, T. and Ferguson, D.F.: "Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More". Prentice Hall, 2005
29. Forrest, B.: "Google Deprecates Their SOAP Search API". O'Reilly Radar, December 18, 2006. http: //radar.oreilly.com/archives/2006/12/google_depreciates_SOAP_API.html
30. García, R., Gimeno, J.M.: "Open Platform for Multichannel Media Distribution Management HTML". Position paper, W3C Workshop on the Future of Social Networking, 15-16 January, Barcelona, Spain, 2009
31. García, R., Gil, R.: "A Web Ontology for Copyright Contracts Management". International Journal of Electronic Commerce, Vol. 12, No. 4, pp. 103-117, 2008
32. García, R., Tsinaraki, C., Celma, O., Christodoulakis, S.: "Multimedia Content Description using Semantic Web Languages". In Y. Kompatsiaris & P. Hobson (Eds.): Semantic Multimedia and Ontologies: Theory and Applications. Springer, pp. 17-54, 2008
33. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z. DBpedia: "A Nucleus for a Web of Open Data". In: The Semantic Web, ISWC/ASCW'07, LNCS, Vol. 4825, pp. 722-735. Springer, 2008
34. Velleman, E., Velasco, C., Snaprud, M., Burger, D. (Eds.) (2006) Unified Web Evaluation Methodology (UWEM 1.0). Retrieved on November 2008 from: http://www.wabcluster.org/uwem1/ UWEM_1_0.pdf
35. Evaluating Web Sites for Accessibility: Overview. Retrieved on June 2009 from: http://www.w3.org/WAI/eval/conformance.html
36. Web Accessibility Evaluation Tools, Complete List. Retrieved on June 2009 from: http://www.w3.org/WAI/ER/tools/complete
37. Preliminary Review of Web Sites for Accessibility. Retrieved on June 2009 from: http://www.w3.org/WAI/eval/preliminary.html
38. Sullivan, T., Matson, R.: "Barriers to use: usability and content accessibility on the web's most popular sites". In: Proc. of ACM Conference on Universal Usability, pages 139–144, 2000
39. Arrue, M., Vigo, M., Abascal J.: "Quantitative metrics for web accessibility evaluation". In: Proceedings of the 1st Workshop on Web Metrics and Measurement (WMM05), Sydney, Australia, 2005
40. Brajnik, G.: "Automatic testing, page sampling and measuring web accessibility". In: Proceedings of the 23rd Annual Int. Technology and Persons with Disabilities Conference, CSUN'08, Los Angeles, CA, 2008
41. Nielsen, J.: "Usability Engineering". AP Professional. Boston, MA, 1993
42. Jeffries, R., Miller, J.R., Wharton, C., Uyeda, K.M.: "User interface evaluation in the real world: A comparison of four techniques". In: Proceedings ACM CHI'91 Conference, pp. 119-124. New Orleans, LA, 1991

43. Nielsen J, Mack RL.: "Usability inspection methods". John Wiley and Sons, New York, 1994
44. Nielsen, J., Molich, R.: "Heuristic evaluation of user interfaces". In: Proc. of the SIGCHI conference on human factors in computing systems. ACM Press, 1990
45. González, M., Masip, L., Granollers, T., Oliva, M.: " Quantitative analysis in a heuristic evaluation experiment". Advances in Engineering Software, in press. 2009
46. Nielsen J.: "Severity ratings for usability problems". useit.com: usable information technology, 1995. Available from: http://www.useit.com/papers/heuristic/severityrating.html
47. Tidwell, J.: "Designing Interfaces: Patterns for Effective Interaction Design". O'Reilly, 2005
48. Caldwell, B., Cooper, M., Guarino, L., Vanderheiden, G.: "Web Content Accessibility Guidelines (WCAG) 2.0 Recommendation". Retrieved on June 2009 from: http://www.w3.org/TR/WCAG20/
49. Craig, J., Cooper, M., Pappas, L., Schwerdtfeger, R., Seeman, L.: "Accessible Rich Internet Applications (WAI-ARIA) 1.0 Working draft". Retrieved on June 2009 from: http://www.w3.org/TR/wai-aria/