A Semantic-Web based Framework for Developing Applications to Improve Accessibility in the WWW

Christos Kouroupetroglou
Dept. of Applied Informatics,
University of Macedonia
Thessaloniki, Greece
+30 2310 791604
kourou@teithe.gr

Michail Salampasis
Department of Informatics,
T.E.I. of Thessaloniki
Thessaloniki, 57 400, Greece
+30 2310 791284
cs1msa@it.teithe.gr

Athanasios Manitsaris
Dept. of Applied Informatics,
University of Macedonia
Thessaloniki, Greece
+30 2310 891898
mantis@uom.gr

ABSTRACT
One of the biggest issues the World Wide Web (WWW) community has to overcome is accessibility for all. The rapid expansion of the WWW using problematic web authoring practices, together with the dominance of the desktop metaphor in web page design has raised many WWW accessibility problems for people with disabilities. In this paper we present a what may be termed as a "Semantic Web application framework" which allows different applications to be designed and developed for improving accessibility of the WWW. Apart from the architecture, the tools and the technologies that compose the framework, the key idea of the framework is that it aims at promoting the idea of creating a community of people federating into groups each playing a specific role: *ontology creators* creating concepts using an ontological approach to describe various elements of the WWW, *annotators* using concepts to annotate specific pages, *user-agent developers* creating tools based on the framework, and finally *end-users* (people with disabilities) that use these tools for their benefit. Within the proposed framework, these groups cooperate and interact with each other, having as their ultimate goal the improvement of WWW accessibility.

Categories and Subject Descriptors
H.4.3 [**Communications Applications**]: Information browsers.

General Terms
Design, Measurement, Management.

Keywords
Semantic web, metadata, RDF, voice browser, accessibility, information seeking.

1 INTRODUCTION
This paper presents a generalised application framework, which enables applications for improving accessibility in the WWW to be designed and developed within an open and extensible underlying framework. The framework is based on the idea of the Semantic Web and it could be utilised to provide the basis for developing tools and applications (user agents in more formal terms) for augmenting mobility and information seeking performance in the WWW for people with disabilities. One key objective of the proposed framework is to promote the idea of forming communities of collaborating people each playing specific roles in the framework. In other words the proposed application framework, like any other metadata scheme, signifies community membership [16].

There are many different accessibility and information seeking problems in the WWW depending on the type of user disability, the information need that must be satisfied, the level of expertise of the user, how the web content was created, the information task at hand etc [2][4][21]. In the following paragraphs of this introductory section we will describe accessibility problems when people with disabilities use the WWW. However, it is not our goal to make a thorough discussion of accessibility and other mobility and information access problems in the WWW. Although the framework could be utilised in any situation in which some form of annotation could be beneficially applied to facilitate mobility and information access in general (e.g. visually impaired people, elderly), in the rest of this paper we will focus on accessibility problems of visually impaired (VI) people for two reasons. First because we have implemented a voice web browser based on the suggested application framework [22], therefore we possess experience about this type of disability and associated accessibility problems. This facilitates our need to give some specific examples of how the generalised framework could help the development of applications addressing specific accessibility problems. But also, because visually impaired are probably the best example of web users that their disability (regarding navigating and accessing information from the WWW) may be compensated, if support based on our framework is provided.

After the discussion of accessibility problems, in this introduction we will also shortly describe the framework and how we envisage that it could act as a foundation and an underlying platform for creating tools and applications that address the accessibility issue in the WWW.

Nowadays, web page design is eminently dominated by the desktop metaphor and generally web page authoring uses unreliable, variable and inconsistent authoring practices. The layout of a web page, the use of various fonts, colors, images, and other visual cues convey navigational and semantic information to sighted users. Unfortunately these visual cues are not accessible by people with disabilities such as blind users [6][9]. For example, a list of links appearing in a table in the left side of a web page with different font and background color is immediately associated by a sighted user with the concept of a navigational

menu. On the other hand, a blind user using a voice browser usually understands the same element as a table with links within its cell. He might conceive it as a “navigational menu” concept, but that will happen only after browsing and listening to its contents several times. Apparently this process makes within web page and across document browsing very inefficient and develops high cognitive overhead.

The latter exposes another problem that originates from the misuse of HTML tags from web developers. One of the basic principles in developing accessible web pages is to use HTML tags correctly to make functional mark-up. For example, using an H1 tag to make a certain text appear in bigger fonts without being a level one heading is a wrong but often used technique. Or a H2 tag may appear without H1 for example. Another common use of HTML tags which causes problems is the use of the TABLE tag for layout and lining reasons without having any tabular data within. The visually conveyed information in these cases is missed and sometimes it confuses even more blind users who use screen readers or voice browsers to read them.

Generally this lack of accessibility leads to poor navigation, mobility and consequently inefficient information access for VI users. Harper [9] has introduced the notion of travel and travel objects to draw an analogy between web navigation and travel in the real world, in an effort to improve web accessibility for VI. Although Goble’s contribution served the need for a mobility analysis framework, Yesilada et. al. [28] have suggested that a more systematic foundation is required for engineering tools in a more systematic way that will support mobility.

Most of the blind users today use programs such as generalised screen readers and specialised voice browsers for their web browsing. A basic functionality of these tools is that they serialize a web page into a simple text. This serialization often brings uninteresting parts of a web page (e.g. advertisement banners and other peripheral material) in front of the main content of a web page. This fact disappoints, upsets and disorientates users. To avoid this “noise” of useless information, voice browsers and screen readers provide additional features such as listening to the links or the headings of a web page. These features provide a quicker access to some parts of a web page that are difficult to reach using the simple serialization. However, the misuse of HTML tags sometimes disables their usability and creates additional problems to browsing within a page.

In addition to problems related to within page browsing, blind users do not have the ability to scan quickly a web page in order to attain a digest and a general conception of it. Scanning is one of the most crucial sub-processes of sighted users when seeking for information in the web. Programs may provide some kind of scanning simulation using various techniques of summarization. However, these again depend on how well structured and authored are the web pages.

The discussion above illustrates that the problem of web accessibility for all has many facets, and such as, is the focus of many emerging areas of study. Each area may contribute a little bit (depending on the specific problem) to produce *together with other techniques* an overall efficient web navigation and effective information access for people with disabilities. In that context, one contribution of our application framework is that it portrays the backdrop for the work of different user agents to complete activities that will enhance web accessibility for people with disabilities. Our research effort aims at designing what may be termed as a "Semantic Web application framework" to support the development of accessible WWW applications for all. The framework suggests an architecture that can be generalized and applied in developing WWW applications for many types of accessibility problems. However, the framework as it is presented and discussed in the rest of this paper is focused particularly in the accessibility problems related to the information seeking process of blind users in the web.

A variation of techniques and strategies can be used for information seeking in the WWW. Browsing is one of them and it is the specific strategy that the actual implementation of the SeeBrowser tool that is based on the framework tries to improve. Browsing is separated in across document and within document browsing. The tools that were developed based on the framework, for this specific project, aim in improving and solving problems for both types of browsing.

The proposed application framework delineates an architecture which in our case is instantiated by a set of software tools that we will describe in the following sections. But, it also presupposes a community of people separated into groups, each playing a different role. The first group is *ontology creators* responsible for creating concepts using an ontology editor. The second role is played by *annotators.* Annotators use available concepts to annotate specific web pages, aiming at increasing their accessibility. Third, *user-agent developers* that create tools based on and exploiting the framework. The last group is *end-users* (people with disabilities) that use user-agents such as the voice web browser presented later in this paper. The roles of the groups of people envisioned in our proposed framework will be presented in parallel with the basic tools that compose the application framework. These groups need to interact and cooperate with each other. This cooperation and interaction is another crucial part of the framework and it is also discussed in the paper.

2 A SEMANTIC WEB APPLICATION FRAMEWORK

Before we describe the application framework we review some earlier work that partially inspired us. As already discussed earlier, one of main problems in within document browsing, is the reconstruction of web pages by serialization. This was realized quite early and several solutions for this problem were suggested. Some of these solutions presented by Tagaki & Asawaka [3][25] and Huang & Sundaresan [11] was based on a transcoding server which transforms web pages according to specific annotation and patterns. The transformation process reconstructs the web pages in a way that makes navigation for blind people easier. The transcoding server approach though, had some drawbacks presented by Hanson & Richards [8]. In addition an interesting approach in storing and retrieving annotations is presented in the Annotea project. The concepts of using annotations, client-side architecture and having the annotations stored on a server are fundamental in our framework too.

One of the latest advances in web technologies, the Semantic Web, came as the ideal background to support all of them. The Semantic Web raised a great deal of discussion and many expectations. Marshall & Shipmann [15] present, categorize and discuss the various views and expectations raised by the Semantic Web. Some people see the Semantic web as library cataloguing system for the web. Others hope that it will increase machine awareness of web content, improving this way searching facilities in the current web. One other category uses Semantic Web as a

way for metadata syndication, enabling the communication among various information sources and agents. This latter approach is the one that our framework is closer to. In our research work we envisage the Semantic Web as a metadata layer upon the current WWW, through which user agents will syndicate, interact and collaborate in order to improve accessibility. These metadata can be produced by various sources and can be used by many users and agents for a variety of goals.

At the cornerstone of our application framework are metadata and their manipulation. Our metadata are stored and retrieved in a storage server using RDF/XML formatted files. RDF is the standard used for metadata in the Semantic Web and is a language for describing resources. We use this language for describing web pages in a way that will help to improve their accessibility. Storing and retrieving metadata from a storage server, allows different users to contribute to the development of metadata at the same time, thus forming a community of users contributing in developing the Semantic Web.

A key element in describing resources in the semantic web is the vocabulary used for this description. The vocabulary can vary depending on the goal of the description. There are already many vocabularies such as DC, Foaf, etc, that can describe various resources in various ways. For the description of a web page in order to improve its accessibility by blind users there are specific needs that were not covered by any of the existing vocabularies. To satisfy these needs we use an ontological approach of OWL (Web Ontology Language) to create a vocabulary especially for them. This solution was preferred among other because is one of the most commonly followed in the Semantic Web nowadays.

Our voice web browser finally utilizes the two previous characteristics, vocabulary and metadata in RDF. All three of them together are the key points of our framework. Figure 1 illustrates an overview of the framework and the relationship between the three key tools used in our framework. SeEBrowser finally uses the outcome from both the annotation tool, which are the metadata, and from ONAR, which is the vocabulary used for the annotation.

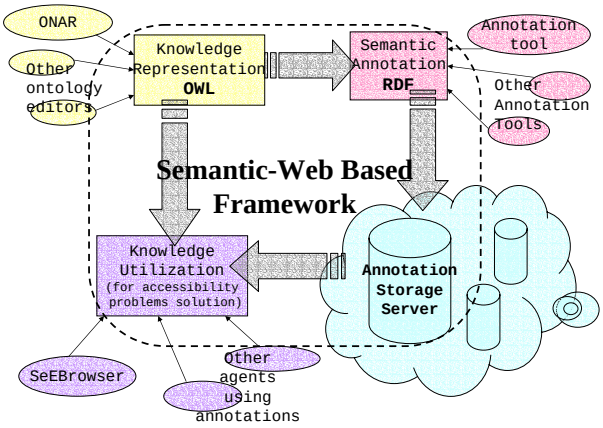


Figure 1 : The suggested framework scheme

Of course, it should be said that the framework could be generalized, as many different uses of metadata could be applied. In our research so far, we have investigated just one of them (i.e. improve information seeking for blind people in the WWW). Based on the annotation mechanism using metadata combined with the vocabulary, the browser provides to the blind users with a set of browsing shortcuts to the previously annotated elements. This mechanism according to the findings of our preliminary experimental tests with a set of blind users using our voice web browser could be quite useful for them [22].

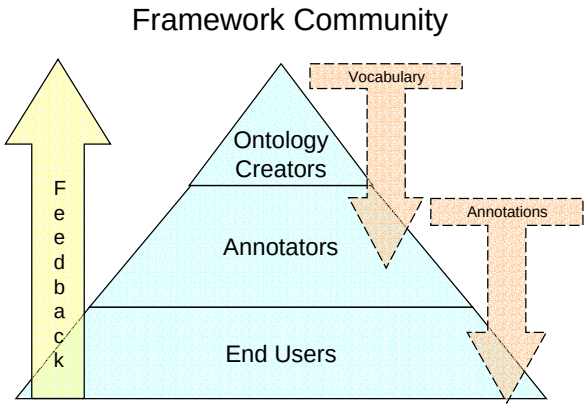


Figure 2 : Framework community diagram

The feedback provided by end-users apart from the SeEBrowser development group was also useful for the annotators' group, because it suggested changes in annotations, better descriptions for the annotations etc. Annotators are one of the groups in the community model suggested by the application framework (see Figure 2).

Their role is to annotate web pages for blind users. People in this group can be either related to web authoring process such as web developers, designers etc. or related to the blind users group such as teachers in special schools for blind people. All of them together could contribute with their annotations to web pages and create an extensible layer of annotations over the existing web.

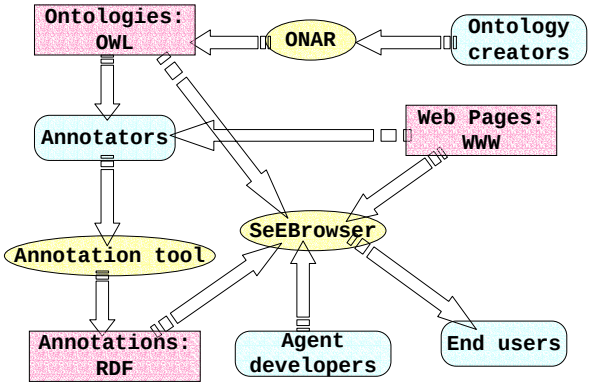


Figure 3 : Framework data flow chart

During the annotation process annotators realize shortcomings in the vocabulary they are using and suggest changes in it. The group taking this feedback is the vocabulary developers and they are responsible for developing vocabularies according to users and annotators needs. This group of people can consist of

knowledge professionals, annotators, and generally people related to knowledge engineering. The process and the connections described in the previous paragraphs are shown in Figure 3 which shows the data flow for the specific instantiation of the framework in our project.

3 OWL ONTOLOGY

One of the key points in our application framework is the vocabulary used for producing the annotations. The vocabulary is described as a set of entities, properties and relationships between ontologies in an OWL file. OWL is a language used for declaring and describing ontologies. The ontologies can depict information systems using classes, properties and relations between classes. Our information system which is modelled is the WWW and the web pages that exist in it together with their key visual and other elements that are often used by sighted users. A menu for example can be a class that describes the entity of a menu in a web page and could have a property named "number of items". In the following section we will demonstrate how the concepts described as classes in one ontology are instantiated in web pages.

In the specific instantiation of the proposed application framework we have developed, an ontology editor called ONAR [27]. ONAR provides a GUI where the knowledge engineer-vocabulary creator can easily create classes, relations and assign properties. All of them are presented in a graphical way as seen in Figure 4. The main advantage of ONAR is that allows users who don't know the OWL language to create ontologies easily. In addition, it is easier to understand an ontology developed by another person when you see it in a diagram, such as this shown in Figure 4, than in an OWL file. This encourages the collaboration amongst vocabulary developers having as a common goal the production of a final ontological vocabulary.

ONAR in its initial form was designed to process ontologies locally on a user's computer. This would mean an additional cost of workload and communication between users when needed to update or review an ontology developed by someone else. Needless to say that there would be also synchronization problems if someone was developing an ontology in parallel with someone else. To avoid all this confusion we enhanced ONAR with an additional feature of downloading and uploading ontologies in the web using a web service. Every user has a username and uses this in order to upload his or her ontologies. In addition he can assign a group of users that can update the current ontology. With this scheme the network of vocabulary creators can either develop an ontology on their own or cooperate with other users in the development. However, when using the ontologies for the annotation process there was the need to have a certain user whose ontologies would be the result of all the suggestions and discussions in the development process. To achieve this we created a specific user with name * whose ontologies are final products from the various collaborations and suggestions. This doesn't exclude ontologies developed by various users or groups to be used too but ontologies by user * are more official than any other.

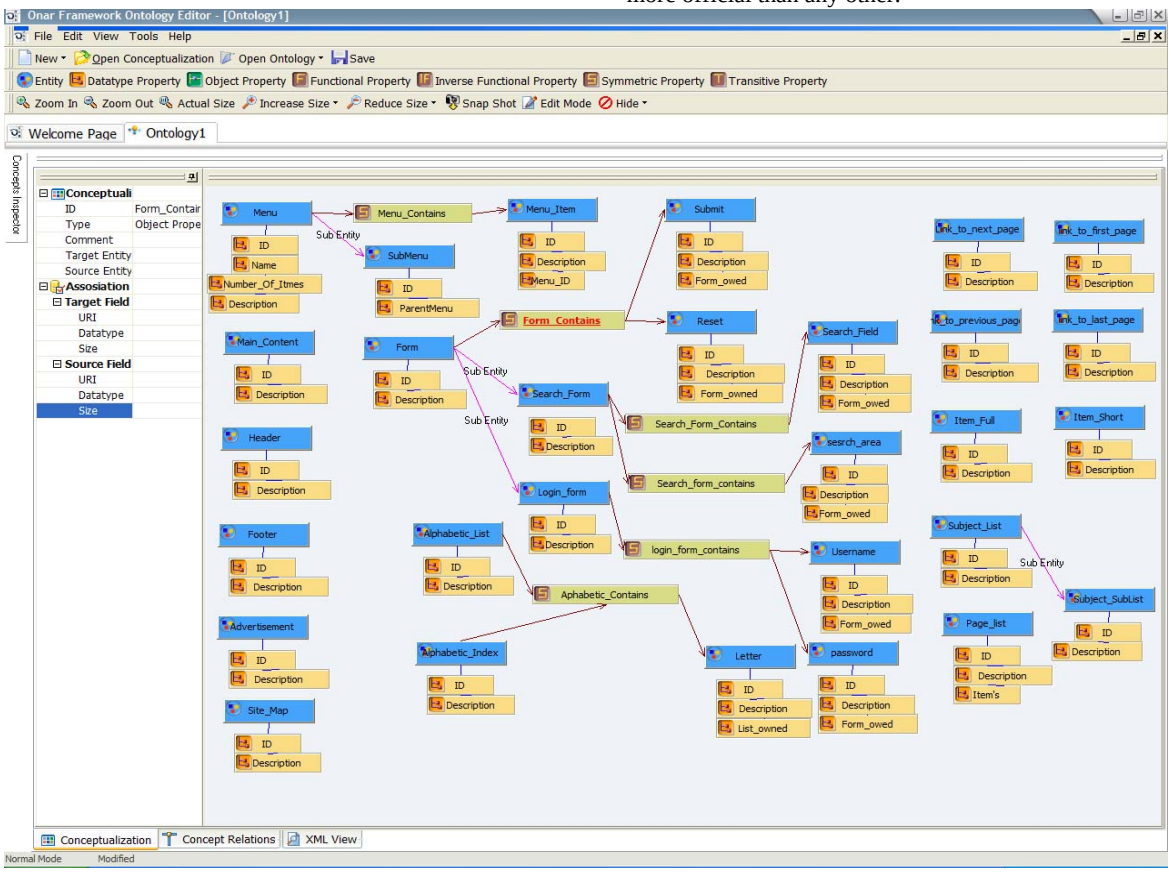


Figure 4 : Screenshot of ONAR representing an ontology graph

3.1 OWL

The vocabulary issue is in general a knowledge representation issue. The choice of OWL in our framework was based first of all on the fact that it is a standard which is used widely in the Semantic Web society. This makes it easier for anyone who wants to contribute in the development of a vocabulary to do so having a common language for communication. In addition ontologies in OWL can serve many different purposes and apart from the ontologies developed for our purpose, anyone could develop a similar ontology for other purposes such as improving accessibility of elderly or people with dyslexia. So, the use of OWL allows extensibility of our framework.

In addition, OWL leaves the knowledge engineer free to construct many kinds of relationships apart from the standard types of relationships. This freedom though, comes with the cost of producing an ontology that might be perceived differently by different annotators. These problems of misinterpretation weren't strong enough to prevent us from using this freedom to create our kind of relationships. For example, in our ontology there is a class called "menu" and another one called "menu item". These two are connected with a relationship named "menu contains". Similar relationship are the "form contains" and the "result list contains". Using "contains" as a part of the relationship's name is a naming convention in our relationships. This way, we could create several different relationships that all have the same meaning and can be used in a certain way by the agent. In our browser for example when the user reaches an annotated web page and listens to the annotated elements found there, elements that are contained within other container type elements are excluded from this initial list. When later the user reaches a container element he can listen the annotated elements found within this container element.

The vocabulary we developed in our SeeBrowser project aims in describing elements in a web page that help blind users to move faster and more efficient within a page and also across pages especially of a certain site. Yesilada et. al. [28] have already presented such a set of elements on web pages. In our vocabulary there is a set of classes with the appropriate properties and relationships that describe many of these elements. This means that we have classes that describe various way points such as menus, headings, sections, banners, advertisements, links to specific places (i.e. site map page, home page, index page) etc. In addition there are classes that describe elements that are widely used in reference pages when searching for information such as subject list, alphabetic list of items, short descriptions of items, elaborate descriptions of items, navigation links within multiple result pages such next, previous, first and last page etc. Another part which is under development describes specific elements widely used in portal sites such us, search box, web directory, login form, weather box, news section, etc. Finally we plan to develop the vocabulary with even more classes especially for educational sites.

4 ANNOTATIONS

Annotations are the second key element of our framework. They are produced in the form of RDF/XML files by an annotation tool. Annotations are stored on an annotation storage server (see Figure 1). In this section we will discuss issues related to the structure of annotation files, architecture of the annotation storage server, and the role of annotators. Annotators play the role of the middleware group between vocabulary creators and end-users and user agent developers. They are also closer to end users and therefore they have a very important and crucial role in our framework.

4.1 Structure of annotation files

Before explaining the structure of the annotation files we should make a brief introduction to RDF. RDF is based on statements that are formed in triples (Subject, predicate, object). Having this in mind, we can say that a subject is a resource a predicate is a property name for this resource and the object can be either another resource or a literal as a value of the property. This means that an object of a statement can be a subject in another statement so that we can have a series of statements in a chain. A usual presentation of an RDF file is a directed graph where resources are presented with ellipses, predicates with arrows and literals with rectangles. A typical graph can be seen in Figure 5.

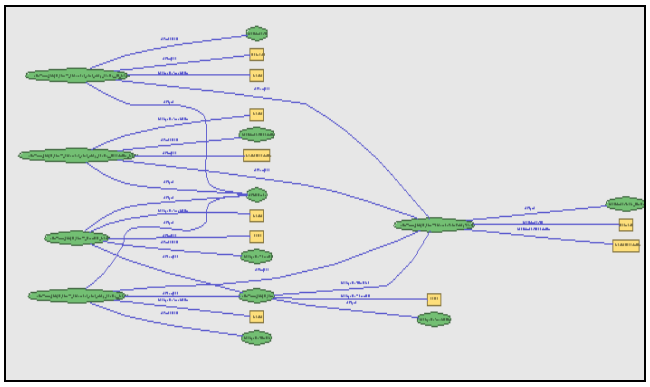


Figure 5 : RDF Graph of an annotation file

For the particular application we investigated as a case study of our application framework (i.e. develop a voice web browser for blind people), we use an xml namespace called "SeESyntax" that includes the schema according to which our files are produced. The second namespace called "SeEBrowser" points to a vocabulary for describing a web page which is an OWL file as seen in the previous section. Having defined these two namespaces, the typical structure of an annotation file in our application is as follows:

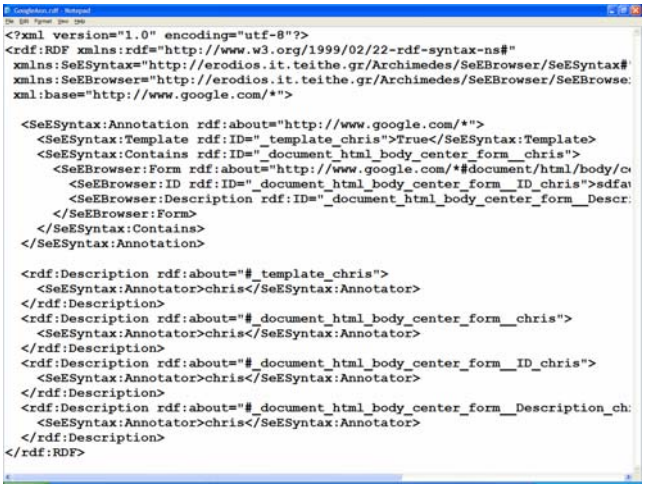
```
<rdf:RDF xmlns:rdf="[RDF namespace]"
xmlns:SeESyntax="[SeESyntax namespace URL]"
xmlns:SeEBrowser="[SeEBrowser namespace URL]"
xml:base="[URL of annotated page]">
<SeESyntax:Annotation rdf:about="[URL of
annotated page]">
  <SeESyntax:Template>False</SeESyntax:Template>
  <SeESyntax:Contains>
    <SeEBrowser:[OWL Class] rdf:about="[URL +
XPATH of the annotated DOM element]">
      <SeEBrowser:[OWL Class property]>[value]
      </SeEBrowser: [OWL Class property]>
      <SeEBrowser:[OWL Class property]>[value]
      </SeEBrowser: [OWL Class property]>
      . . .
    </SeEBrowser:[OWL Class]>
  </SeESyntax:Contains>
  <SeESyntax:Contains>
  . . .
  </SeESyntax:Contains>
. . .
</SeESyntax:Annotation>
</rdf:RDF>
```

A file structured like this can describe various elements of a web page using the XPATH of the HTML element one wants to

annotate. For example, an annotator assigns the concept of a menu in a certain TABLE element on a web page and similarly assigns values to its properties defined in the OWL file. In a similar way another part within a DIV element in an HTML file, can instantiate the concept of "main content" and so on. This is done using the "SeESyntax:Contains" nodes.

The node "SeESyntax:Template" is a property of the SeESyntax:Annotation class and is used in order to declare whether the URL of the annotated page is a template for other URL's too or not. This is done in order to reduce the workload and size of annotations produced when a certain page is similar to others. Pages within a site are quite common to follow a certain layout as a template for their design. When annotating one of them we can use the Template property and in combination with a regular expression instead of the actual URL we describe not just one but a set of web pages described by this expression. This mechanism provides a way of semi-automatic annotation of web pages, which is crucial when annotating a large amount of pages.

As already discussed, annotations could be produced by many different annotators using different vocabularies. The capability of using different vocabularies is addressed in our framework by the capability of defining different vocabularies. This is done by the xml namespace declaration where every annotator can choose and use any preferred vocabulary.



```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:SeESyntax="http://erodios.it.teithe.gr/Archimedes/SeEBrowser/SeESyntax#"
 xmlns:SeEBrowser="http://erodios.it.teithe.gr/Archimedes/SeEBrowser/SeEBrowse
 xml:base="http://www.google.com/*">

  <SeESyntax:Annotation rdf:about="http://www.google.com/*">
    <SeESyntax:Template rdf:ID="_template_chris">True</SeESyntax:Template>
    <SeESyntax:Contains rdf:ID="_document_html_body_center_form__chris">
      <SeEBrowser:Form rdf:about="http://www.google.com/*#document/html/body/c
        <SeEBrowser:ID rdf:ID="_document_html_body_center_form__ID_chris">sdfa
        <SeEBrowser:Description rdf:ID="_document_html_body_center_form__Descr
      </SeEBrowser:Form>
    </SeESyntax:Contains>
  </SeESyntax:Annotation>

  <rdf:Description rdf:about="#_template_chris">
    <SeESyntax:Annotator>chris</SeESyntax:Annotator>
  </rdf:Description>
  <rdf:Description rdf:about="#_document_html_body_center_form__chris">
    <SeESyntax:Annotator>chris</SeESyntax:Annotator>
  </rdf:Description>
  <rdf:Description rdf:about="#_document_html_body_center_form__ID_chris">
    <SeESyntax:Annotator>chris</SeESyntax:Annotator>
  </rdf:Description>
  <rdf:Description rdf:about="#_document_html_body_center_form__Description_ch
    <SeESyntax:Annotator>chris</SeESyntax:Annotator>
  </rdf:Description>
</rdf:RDF>
```

Figure 6 : A sample of an annotation file

This scheme however is problematic when annotations from different annotators will be uploaded to the server. Annotations are stored all together using a specific API that manages them using an RDBMS. This means that statements of various users about the same resource might create conflicts. To avoid this confusion we use a mechanism provided by RDF and is called reification. Reification is used when someone wants to make a statement about a statement. This is done by using a certain set of classes and attributes that convert a statement to a resource itself so it can be used as subject in other statements. To achieve this we use the abbreviated syntax for reification. This means that we assign an rdf:ID property to each statement's predicate and later using this rdf:ID we can assert the reification statements. Consequently, the structure of the file changes to what is shown in Figure 6. This way not only the problem is solved using RDF mechanisms, but there is also the possibility to use annotations synthetically by many users.

4.2 The annotation tool

Annotation files are produced using specially designed software. Figure 7 illustrates the user interface of the annotation program that is separated into three main areas. The main area in the center presents to the annotator the web page to be annotated. On the left hand side a tree view represents the DOM tree of the web page. When the annotator selects an element from the DOM tree the corresponding area of the element is highlighted in the page view. On the other side there is a list of the OWL classes available in the vocabulary. When a user right-click on an element from the DOM tree, this list appears in a popup menu and the annotator can select the concept that s/he wants to assign to the element. After the selection, a series of dialog boxes appear asking the user to input values to the properties of the class. Repeating these steps annotators produce the annotation file for a web page. Before saving or uploading the file to the storage server the tool asks them to input their username so that the reification statements can be produced. Finally it also asks them whether the page they annotated is a template for other pages or not. If yes then the user has to input the regular expression that will describe the set URL's for the annotated pages.

4.3 Storage Server

The storage server is based on the idea of Annotea project [1] and exploits some of its advantages. First of all the RDF annotations are stored on a relational database using mySQL as RDBMS. The Jena API [13] is used for managing the database. Jena API provides the developer with the ability to work with an RDBMS as an RDF database using RDQL commands instead of SQL. RDQL is a language similar to SQL but especially developed for selecting and presenting RDF graphs stored in RDF databases.

One of the advantages of the Annotea project is that the communication and management of annotations in a server can be done through simple HTTP POST and GET commands. We also used a similar protocol for our communication with the annotation server.

In particular a user requests the annotations for a URL with an HTTP POST command. If there are many annotators that uploaded annotations for the specific URL, then the response is a list of them and there must be a second HTTP POST command accompanied by an annotator's name in order to get the annotations from a specific annotator. Otherwise the annotations are sent directly as a response to the first POST command. When the server searches for a URL matching the requested it also checks if the requested URL matches with regular expressions existing in template descriptions.

The upload process is also similar. The server keeps a table of annotators that can upload annotation to the server and when a user uploads a file the username and the password kept in the server must be also sent through an HTTP GET command to be checked in the server. This requires annotators to be inserted in the annotator's table beforehand by the annotation storage server administrator. This is done for authorization purposes so that there is a control on who uploads annotations.

For the purpose of our project we set up a specific storage sever that we are using. The architecture though of this scheme is extensible so that we could also have a network of storage servers. This means that this network could become a second semantic layer upon the current web. Agent developers can then take advantage of this layer and use their metadata in any way they like.

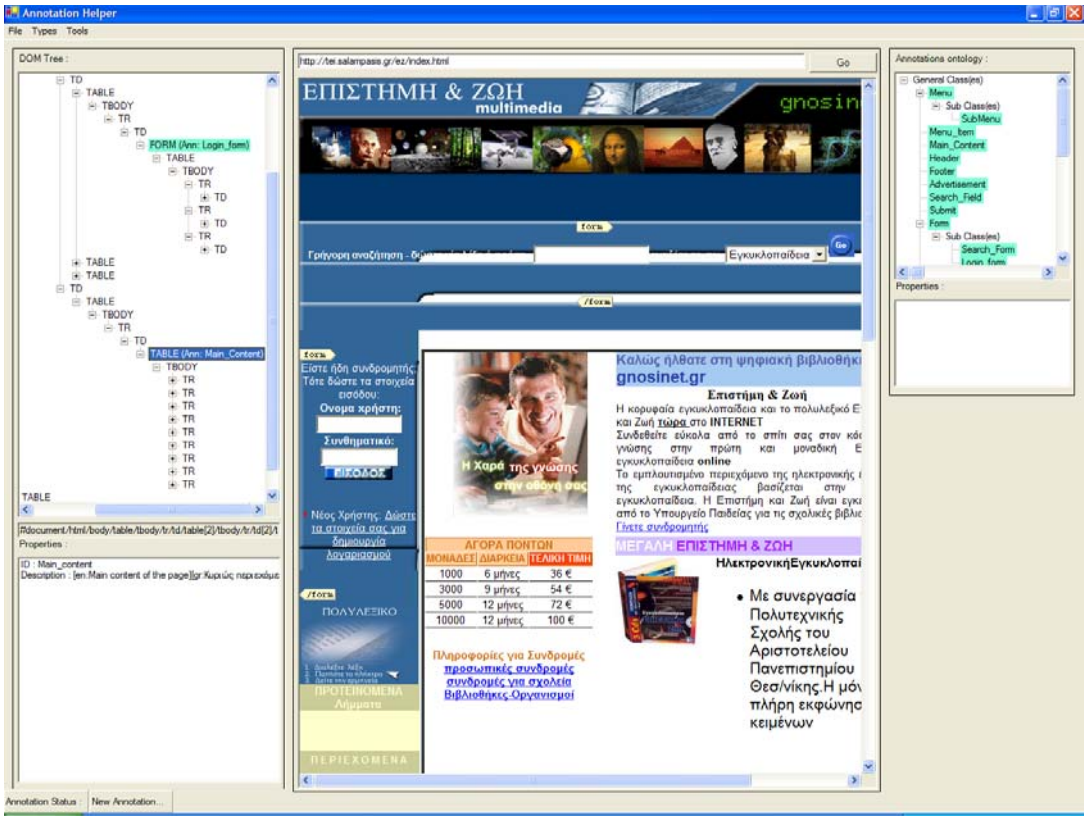


Figure 7 : Screenshot of the annotation tool

4.4 The role of an annotator

The authorization process is required because it is essential to control the annotations uploaded in the server. We need to know if an annotator produces invalid or misguiding annotations and isolate or delete them. End users depend on good annotations, so annotators need to have certain level of commitment and reliability.

The latter shows that this group is very important in the framework. An annotator should have some primitive knowledge of HTML in order to understand the DOM tree structure and use it correctly. The other important part of the annotation process is the vocabulary. Names of the classes, their description, role and relationships should be well understood before the annotation process begins. Misinterpretation of concepts in the vocabulary could lead to false or incomplete annotations. Finally annotators should have a clear and complete overview of the web pages that annotate in order to know their structure, layout and navigational aids that they provide.

Another very important aspect of the annotation process is the purpose of the annotation. The annotator should know where and how these annotations would be used in order to achieve a better description of the page. Knowing the needs of end users the annotations may respond better to their needs. It is similar to the situation when one must describe a building to an architect and also to a friend without particular knowledge of the subject. First of all, the vocabulary in the first case should be more technical and specific where in the other case simpler. In addition the architect will need more details in the description such as exact sizes and places where the other person will be satisfied even with a general description of the building.

In an analogous way, annotators in our example should know what problems a blind user faces when browsing in the WWW, in order to provide usable solutions through the annotations. Consequently, interaction between these two groups is necessary. Certainly the closer an annotator is to the end users group, it is more likely he can produce a more effective annotation. Annotators also provide feedback to vocabulary developers in order to transfer to them needs of blind users that are not satisfied by the current vocabulary or possible misinterpretations of the concepts defined in a vocabulary.

To sum up, annotators as an intermediate group, between vocabulary developers and end-users take feedback from end-users and provide feedback to vocabulary developers regarding the *expressiveness, correctness and appropriateness* of the vocabulary. Sometimes it is even better if an annotator plays also the role of vocabulary creator because he can solve annotation problems related to vocabulary. There is also the possibility for end-users to be annotators. However, in our case blind users that want to annotate pages need to have an even better understanding of HTML and they also need to have a sufficient browsing experience with the pages to annotate.

5 SEEBROWSER

SeEBrowser is the final part of the framework and is the tool that utilizes the result of all other tools and groups of people for end-users benefit. It should be clear-cut that in our framework the two other components that have been already discussed, i.e. the ONAR ontology editor and the annotation tool can be used in any condition and for developing any type of Semantic-Web application. In this section we will shortly describe the basic

functions of SeEBrowser and we will discuss the findings from a preliminary usability test.

5.1 Basic features of SeEBrowser

SeEBrowser uses SAPI5 compatible TTS engines and voices. Especially for the Greek language it uses the "Demosthenes" TTS engine [30]. Using SAPI5 compatible voices means that users should be able to change their preferred voice. This is done using a voice profile control panel where users can configure the preferred voices and other configurable aspects of speech (e.g. adjusting voice rate).

Similar to other web browsers, SeEBrowser users can insert a URL to browse, follow a link within a page, go back and forth in visited pages and go to the home page. There is also a search text feature allowing users to move directly to instances of a specific text within a page. Finally there is a bookmarks feature that allows users to save favourite URL addresses in a list. This feature however has been adapted slightly to blind users' needs. Apart from the URL and the title it can also store the current position of the reading cursor. This is later used to transfer the user directly to the specific position when opening the page from the bookmarks list.

5.2 Browsing within a web page

How the reading cursor moves, depends on how the user browses within a page. Users listen to the web page fragmented depending on the combination of which browsing and speaking mode is selected. Based on these two modes a web page is decomposed in two levels in order to be separated into the fragments to be browsed.

The first level is the browsing mode selected. The browsing mode defines whether the user will browse either the whole text of a web page or parts of it (e.g. links only). In particular the user can select either to browse the whole page or its links, headings or forms. In each of the later cases a list of the HTML elements to be browsed is formed. It is also important to say that when returning from a specific collection of elements to browsing the whole page, the "cursor" automatically moves to the corresponding place next to the last element browsed by the previous mode.

The second level is the speaking mode selected. Here the text of each element is further separated either in paragraphs, sentences or words. This way the user pressing the Up and Down arrow keys can listen the selected fragment word by word, sentence by sentence or paragraph by paragraph. The combination of browsing and speaking modes provides the user with a variety of possible ways to browse a page according to his/her needs.

5.3 The use of annotations

As already pointed out a distinctive feature of SeEBrowser is the use of the annotations produced by the process described in the previous sections. There can be various uses of annotations but the one implemented in SeEBrowser in this phase of the research aims in improving browsing as an information seeking strategy. This means that we aim in improving both browsing across pages and within a page in order to make the information seeking process more efficient and effective. The feature provided by our browser for this purpose is the shortcuts to the annotated elements.

This feature aims basically in improving the browsing within a web page by simulating the layout scanning process of a sighted user. When a user browses an annotated web page he can listen to an overview of the page based on the annotated elements that exist in it, by pressing Alt+I. Then using the Alt+Up or Down arrows can move to each of them and start browsing its content. When for example he visits the page show in Figure 7 he listens that there are a main content area, a search box, a login form and various other elements. Then by pressing Alt+Down arrow he can move to the element he wants. If for example he wants to reach the main content area he has to browse through the elements and move to it. Once he hears the message "You are now in the main content area" he can navigate and listen to it using the up and down arrows. This reduces the time needed to reach the specific point if he was using the simple navigation within the page. In this case he should have "travelled" through every single bit of peripheral uninteresting information in the page and then reach the main content area. This way the overview presented in the beginning offers a set of choices of starting points to reading the content of a page similarly to what a sighted user does when visiting a web page.

SeEBrowser using browsing shortcuts provides also faster navigation through various elements of a page. Consider for example an end-user who starts reading the main content and judges it as not relevant; s/he might need to move directly and use a navigational aid such as a menu in the page. Using SeEBrowser's browsing shortcuts feature s/he can move to the desired element faster by simply navigating the annotated elements list. Without this feature the end-user would have used the simple navigation within the text in order to find a specific point, possibly a phrase that would signal the existence of a menu. Many blind users memorize distances in paragraphs or links for these elements in order to find them later using the start of the page as a landmark. Both ways are more time and effort consuming than SeEBrowser's browsing shortcuts utility.

The mechanism hidden behind this feature is the annotations and their properties. Every class in the ontology has two standard properties. The first, named "ID", identifies uniquely each annotated element from any other in the page. The second named "Description" contains a short description about the annotated element and is heard when the user reaches the specific element.

Furthermore, the groups of relationships presented in the OWL Ontology section allow some special management for some of the annotated elements. In particular the "contains" group of relationships indicates to the browser the existence of a hierarchy of classes. In our case, when a user listens to annotated elements found in a page, some of the annotated elements are excluded from the initial list. These are the elements that are contained within other container elements (e.g. items within a navigational menu). Users can find and hear a list of them only if they reach the corresponding container element. For example the blind user hears that there is a menu in the web page, but only after reaching the menu element and pressing Alt+I again s/he listens that there are 7 menu items within the certain menu element. This allows the annotator to create a quite detailed description of a web page with a controlled level of granularity. It allows also blind users not to be overloaded by hearing elements that are not useful at a specific browsing moment. These elements will be hidden until discovered by the blind user while browsing within the page.

In the current phase of the research the first aim of the vocabulary and the annotations in pages is to improve browsing within a page. However, this improvement of browsing within a page sometimes leads to improvement of across document browsing. For example, pointing to a menu and describing the destination

web page of menu items, makes browsing within a site quite easier. In addition, when examining a search engine's result list the annotations of links to next, previous, first and last page improves a lot the browsing in it. The main benefit from these annotations is that they provide a quicker way of reaching important parts of a web page instead of having to listen to useless information to reach it.

5.4 Preliminary evaluation and experts testing

SeEBrowser was tested by blind users in an experiment presented in [22]. The usability evaluation indicated that the browser was found quite usable, easy to learn and especially the shortcuts feature rated as very helpful by all users.

Further examination of the log files led to some more findings. As seen in Figure 8 the percentage of keystrokes used when using annotations shows that most of the keystrokes are for movement within pages. Excluding these keystrokes (Up and Down arrows) we can have an analysis on the rest of the keystrokes. For the case of using the annotation that we see in Figure 9 we can say that most of the movement across pages are done by following links within pages and rarely going back to already visited pages. This might be come as a result of the structure of the experimental site which was quite simple. The use though, of annotation related keystrokes, shows that they were used almost in every page visited since the Alt+I and Alt+Down percentages are similar to those of EnterLink.

Another analysis on the speaking and browsing modes showed that most users had selected the combination of the whole page as browsing mode and paragraphs as speaking mode. There are very few cases of use of links browsing mode and sentences speaking mode. This could be caused by either the fact that the users weren't too experienced with the application or because of the site's construction that encouraged this combination. It is also important to say that at the time of the experiment the browsing modes for headings and forms were not implemented.

After further development of the application, we gave the tool for experimental use to a number of experienced blind users that would provide feedback through interviews. They were asked to use the tool both in a not annotated site and in an encyclopaedia site annotated for the next experiment. Summing up the feedback from the interviews there are some very important conclusions.

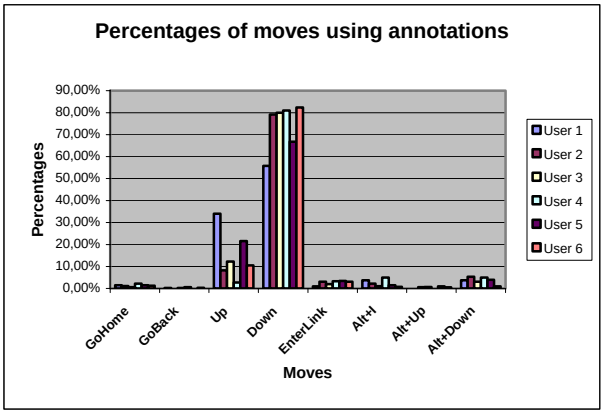


Figure 8 : Percentages of moves when using annotations

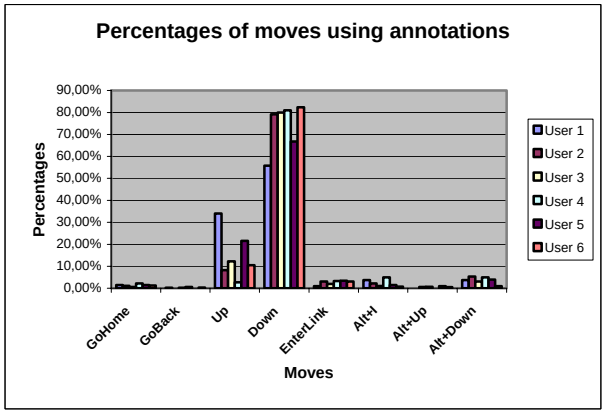


Figure 9 : Average percentage of moves when using annotations

First of all, the modification in the bookmarks feature was welcomed and judged as very helpful. There was also a suggestion to implement the "history browsing" (going back and for in pages already visited) in a similar way. This means that when they move back to the previous page visited they are also transferred directly to the place (paragraph, sentence or word) they followed the link from.

Second the use of annotations in a more realistic environment such as the encyclopaedia was found much more helpful than in the first experiments environment. The use of relationships and annotated elements within other using the "contains" group of relationships was also rated positively.

A disadvantage of SeEBrowser was the lack of particular handling for tables containing tabular data. In our solution a table is read row by row and each cell in a row is presented as a paragraph. This leads to disorientation when navigating large tables. Currently, our solution deals with tables when used for layout because they can be annotated and convey their visual information to blind users too. The solution provided for tables containing tabular data was thought to be sufficient but the feedback we got is now suggesting that it's not. This problem has been investigated by Filepp et. al. [4] and Yesilada et. al. [28] and there are already solutions suggested and could be adapted to our application. Information seeking often has to do with reading tables so this is a crucial part that needs improvement on our browser and it is rated as high priority in our further development.

Finally there was a suggestion for a notepad like feature. In particular they requested a mechanism that would help them in gathering information while seeking. The idea is that when the user finds an interesting part in a page he could mark it and move it directly to a notepad. After gathering an amount of information there, he could save it in a separate text file and further process it later.

6 FURTHER RESEARCH

6.1 Domain specific vocabularies

The next stages of our research include development of domain specific vocabularies and annotation of pages using them in order to improve even more the across document browsing. This can be done by using these annotations for providing functions similar to the scanning function of sighted user which is one of the basic functions in information seeking.

6.2 Automatic annotations

Our application framework depends heavily on the annotations produced by annotators. These annotations up to now are produced mainly by users that contribute them to our framework. It is easily understood that the large amount of information existing in the web and its daily update makes this job quite difficult even for a large network of human annotators.

To deal with this problem there are already thoughts and research in the field of automatic and semi-automatic annotation. The field can be separated in web content and web structure mining. In the field of web content mining there is already research from Huang & Sundaresan [12] and Mukherjee et. al. [17]. In the part of web structure mining there are interesting approaches by Pontelli & Son [19] and Kottapally et. al. [14]. We are currently investigating solutions in structure mining since it seems that in web page design there are specific patterns that are repeated numerously. One of them, that is widely used, is the desktop metaphor. If there can be a set of rules that could find patterns that usually present elements described in our vocabulary then the annotation could be done almost automatically. There will still be human interference in the annotation process to confirm the suggested annotations or correct them but a great load of work would be done automatically.

Up to now none of the suggested solutions for automatic web content or web structure mining is 100% successful to all pages and there are doubts if there will ever be such a solution. There are however, a number of solutions that present quite important percentages of success in specific domain web pages such as in [7] and [30]. This is a good indication that there can be solutions in other domains too.

6.3 Other uses of annotations

Another area of research to be followed is other possible uses of annotations apart the shortcuts functionality. An approach to this could be the formation of profiles that would reconstruct a web page according user needs. These profiles could either be constructed by users themselves, by annotators or even better by machines through semantic web usage mining.

The latter one is a field of research that is now starting to be exploited. For our project there is already a logging function enhanced in the browser that captures movement of users within and across document. These log files could provide valuable information especially for annotated web pages. They could provide users with navigational guidance based on previous visits and browsing in pages. A similar approach in web usage mining is presented by Spiliopoulou [24]. In our research similar techniques could be used on log files from navigation within annotated pages. In general the layer of annotations created by annotators could provide valuable feedback when used by blind users.

6.4 Discussion

Summing up, the paper presented a framework that is based on the Semantic Web idea and may contribute to improving accessibility in the WWW. The framework utilizes various existing standards of Semantic Web such as OWL and RDF. But also it goes beyond the simple utilization of standards, by suggesting an architecture and an application framework that could be generalized to virtually any (Semantic Web) application.

We have presented a particular implementation of the framework that uses an ontology editor and a graphical annotation tool. These two components can be used in any condition and for developing any type of Semantic-Web application. As an example of such application we have presented a specialized voice web browser called SeEBrowser which particularly aims to make browsing blind users more efficient using browsing shortcuts.

However, the most important idea of the proposed framework is that it promotes and encourages the creation of a community that will work together having as their goal the improvement of accessibility. The framework also provides all the necessary tools to facilitate collaboration. This community consists of groups of people each having a specific role in it. Anyone willing to help can contribute from his or her own part, as an ontology creator, annotator, application developer or even end user. The power of this community is its independency and freedom from the current web authoring community. Our community is not tightly connected to the web authoring society, which is quite large and difficult to educate in accessibility issues. However, it can work independently upon the products of the web authoring society.

7 ACKNOWLEDGEMENTS

The research project is funded by the E.U. and the Greek Ministry of Education under the research program “Archimedes”[23]. We would also like to thank Dimitris Tektonidis for his contribution to the project with the development of ONAR and Marios Chatzidimitriou for the development of the annotation storage server software.

8 REFERENCES

[1] Annotea project http://www.w3.org/2001/Annotea/

[2] Asakawa, C. "What's the web like if you can't see it?", In: W4A '05: Proceedings of the 2005 International Cross-Disciplinary Workshop on Web Accessibility (W4A), 2005, 1-8

[3] Asakawa, C. and Takagi, H. "Annotation-based transcoding for nonvisual web access", In: Proceedings of the fourth international ACM conference on Assistive technologies, 2000, 172-179

[4] Coy, J. "A commercial perspective on universal access and assistive technology: towards implementation", Universal Access in the Information Society, 2(3), 2003, 207-214.

[5] Filepp, R., Challenger, J. and Rosu, D. "Improving the accessibility of aurally rendered HTML tables", In: Proceedings of the fifth international ACM conference on Assistive technologies, 2002, 9-16

[6] Goble, C., Harper, S. and Stevens, R. "The travails of visually impaired web travelers", In: Proceedings of the eleventh ACM on Hypertext and hypermedia, 2000, 1-10

[7] Gupta, S. and Kaiser, G. "Extracting content from accessible web pages", In: W4A '05: Proceedings of the 2005 International Cross-Disciplinary Workshop on Web Accessibility (W4A), 2005, 26-30

[8] Hanson V.L., R. J. "Achieving a more usable WorldWide Web", Behaviour and Information Technology, 24 (3), 2005, 231-246.

[9] Harper, S. Web Mobility for Visually Impaired Surfers. PhD thesis, The University of Manchester, 2001.

[10] Harper, S., Goble, C. and Stevens, R. "A pilot study to examine the mobility problems of visually impaired users travelling the web", SIGCAPH Comput. Phys. Handicap., (68), 2000, 10-19.

[11] Huang, A. W. and Sundaresan, N. "A semantic transcoding system to adapt Web services for users with disabilities", In: Proceedings of the fourth international ACM conference on Assistive technologies, 2000, 156-163

[12] Huang, A. and Sundaresan, N. "Aurora: a conceptual model for Web-content adaptation to support the universal usability of Web-based services", In: Proceedings on the 2000 conference on Universal Usability, 2000, 124-131

[13] Jena Semantic Web Framework http://jena.sourceforge.net/

[14] Kottapally, K., Ngo, C., Reddy, R., Pontelli, E., Son, T. C. and Gillan, D. "Towards the creation of accessibility agents for non-visual navigation of the web", In: Proceedings of the 2003 conference on Universal usability, 2003, 134-141

[15] Mantratzis, C., Orgun, M. and Cassidy, S. "Separating XHTML content from navigation clutter using DOM-structure block analysis", In: HYPERTEXT '05: Proceedings of the sixteenth ACM conference on Hypertext and hypermedia, 2005, 145-147

[16] Marshall, C. C. and Shipman, F. M. "Which semantic web?", In: HYPERTEXT '03: Proceedings of the fourteenth ACM conference on Hypertext and hypermedia, 2003, 57-66

[17] Mukherjee, S., Ramakrishnan, I. and Kifer, M. "Semantic bookmarking for non-visual web access", In: Proceedings of the ACM SIGACCESS conference on Computers and accessibility, 2004, 185-192

[18] Pontelli, Son, Kottapally, Ngo, Reddy and Gillan "A system for automatic structure discovery and reasoning-based navigation of the web", Interacting with Computers, 16 (3), 2004, 451-475.

[19] Pontelli, E. and Son, T. "Planning, reasoning, and agents for non-visual navigation of tables and frames", In: Proceedings of the fifth international ACM conference on Assistive technologies, 2002, 73-80

[20] Ramakrishnan, I., Stent, A. and Yang, G. "Hearsay: enabling audio browsing on hypertext content", In: Proceedings of the 13th international conference on World Wide Web, 2004, 80-89

[21] Richards, J. and Hanson, V. "Web accessibility: a broader view", In: Proceedings of the 13th international conference on World Wide Web, 2004, 72-79

[22] Salampasis, M., Kouroupetroglou, C. and Manitsaris, A. "Semantically enhanced browsing for blind people in the WWW", In: HYPERTEXT '05: Proceedings of the sixteenth ACM conference on Hypertext and hypermedia, 2005, 32-34

[23] SeEBrowser Project (Archimedes) http://erodios.it.teithe.gr/archimedes/English/Index.htm

[24] Spiliopoulou, M. "Web usage mining for Web site evaluation", Commun. ACM, 43 (8), 2000, 127-134.

[25] Takagi, H. and Asakawa, C. "Transcoding proxy for non visual web access", In: Proceedings of the fourth international ACM conference on Assistive technologies. 2000, 164-171

[26] Takagi, H., Asakawa, C., Fukuda, K. and Maeda, J. "Site-wide annotation: reconstructing existing pages to be accessible", In: Proceedings of the fifth international ACM conference on Assistive technologies, 2002, 81-88

[27] Tektonidis D., Bokma. A., Oatley G., Salampasis M. "ONAR: An ontologies-based service oriented application integration framework", In: 1st International Conference on Interoperability of Enterprise Software and Applications, Geneva, Switzerland, 2005

[28] Yesilada, Y., Stevens, R. and Goble, C. "A Foundation for Tool Based Mobility Support for Visually Impaired Web Users." Paper presented at the Proceedings of the twelfth international conference on World Wide Web 2003.

[29] Yesilada, Y., Stevens, R., Goble, C. and Hussein, S. "Rendering tables in audio: the interaction of structure and reading styles", In: Proceedings of the ACM SIGACCESS conference on Computers and accessibility, 2004, 16-23

[30] Xydas G., K. G. "The DEMOSTHeNES Speech Composer", In: 4th ISCA Tutorial and Workshop on Speech Synthesis (SSW4), Perthshire, Scotland ,2001, 167-172