Turning the Mouse into a Semantic Device: The *seMouse* Experience*

Jon Iturrioz, Sergio F. Anzuola, and Oscar Díaz

The Onekin Group
Department of Languages and Computer Systems
University of Basque Country
Pº Manuel de Lardizabal, 1
20.018 San Sebastián (Spain)
{jon.iturrioz, jibfeans, oscar.diaz}@ehu.es

Abstract. The desktop is not foreign to the semantic way that is percolating broad areas of computing. This work reports on the experiences on turning the mouse into a semantic device. The mouse is configured with an ontology, and from then on, this ontology is used to annotate the distinct desktop resources. The ontology plays the role of a clipboard which can be transparently accessed by the file editors to either export (i.e. annotation) or import (i.e. authoring) meta-data. Traditional desktop operations are now re-interpreted and framed by this ontology: copy&paste becomes annotation&authoring, and folder digging becomes property traversal. Being editor-independent, the mouse accounts for portability and maintainability to face the myriad of formats and editors which characterizes current desktops. This paper reports on the functionality, implementation, and user evaluation of this "semantic mouse".

Keywords: Semantic Annotation, Knowledge Management, Metadata and Ontologies.

1 Introduction

Hard-disk drives enjoy an increasing storage capacity that, however, has not come along with a similar improvement on mechanisms that harness this storage power. It is at least dubious whether current desktops have scaled to handle a number of files that doubles or triples the ones a layman held a few years ago. Scalability not only includes fault-tolerance or performance stability, but also the availability of tools that permit the end user to harness this power. The lack of appropriate tools for locating, navigating, relating or sharing bulky file sets are preventing PC users from taking full benefit of their storage power.

Current efforts on enhancing operating systems with automatic meta-data extraction support (e.g. Spotlight for MAC[1] or WinFS for Windows[2]) strive to overcome this

* This paper is an extension of a demostration poster presented at the First Semantic Desktop Workshop.

[1] http://www.apple.com/macosx/features/spotlight/

[2] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwinfs/html/winfs03112004.asp

Y. Sure and J. Domingue (Eds.): ESWC 2006, LNCS 4011, pp. 457–471, 2006.

shortcoming. However, these proposals are format-oriented. That is, the extension of the file (e.g. *.doc .ppt .html*) determines the meta-data to be extracted. If all files belong to the same extension, identical meta-data is extracted from all of them. The extractor does not consider the semantics of the content, but the format of the container. This falls short in a desktop setting where the meta-data to be extracted is highly dependent on the user's mental model. The ontology should not be format-centric but user-centric. But this can be cumbersome if appropriate tooling is missing.

Distinct works strive to provide sophisticated new environment or enhancing current tools for achieving the semantic desktop[17, 18, 2]. However, most often this implies for the user to move to a new editor when annotating (like in *SMORE* [12]), or to learn a new "ontological interface" when files from different formats are edited (like in *SemanticWord [19]*). This can pose important usability issues that refrain the adoption of the semantic desktop.

Based on this observation, this work rather than providing separate tools for annotation & authoring, enhances a popular tool for traditional copy&paste operations: the mouse. This will certainly facilitate user adoption.

To this end, the *semantic mouse* (*seMouse*) is introduced. By clicking on its middle button, *seMouse* exports/imports properties from the *ontology,* regardless of the editor you are working with. It does not matter whether you are working with *Word, Excel, PowerPoint, Adobe Acrobat*, *Netscape*, etc, the "semantic" button is available for annotation/authoring.

The rest of the paper is organized as follows. Section 2 addresses related work. Section 3 introduces *seMouse* through five scenarios, namely, file classification, annotation, authoring, semantic navigation and ontology editing. As previously mentioned, usability is one of the main objectives of our tool, so an evaluation is addressed in section 4. The architecture of the implementation is introduced in section 5. Finally, conclusions are given.

2 Framing the Work

This work aligns with current efforts for desktops to become semantic-aware[9, 3, 17]. The endeavors target different goals, namely, semantic infrastructure (e.g. *Gnowsis* [17]), resource organization (e.g. *Fenfire* [4], *Haystack* [13]), annotation/authoring of resources (e.g. *SemanticWord [19]).*

For the purpose of this paper, approaches to inlaying semantics into desktops can be classified in accordance with the coupling between editing concerns (e.g. spelling, grammar checking, layout) and "ontological" concerns (e.g. authoring, annotation). Some approaches to annotation detach the process of annotation from that of authoring (e.g. *Ont-O-Mat* [6]). Others, like *SMORE* [12], do both (i.e. authoring and annotation) but using *ad-hoc* editors.

By contrast*, SemanticWord [19]* integrates semantic capabilities into *MS Word* editor so that the text of the document can be analyzed and annotated as it is being typed, appearing to the author as a service analogous to *Word's* spelling and grammar checking. Moreover, the *MS Word* GUI is augmented with toolbars that support the annotation process. The use of a popular editor certainly facilitates the introduction of these

Table 1. Tool comparison table

Feature	Ont-O-Mat	Semantic Word	OntoOffice	SMORE	seMouse
coupling	Proprietary editor	Word plug-in	MS Office plug-in	Proprietary editor	Windows plug-in
file-format support	HTML	MS Word documents	MS Office Documents	HTML, mail format	Any type
attachment availability	Yes	Yes	No	No	No
GUI device for semantic interactions	Drag & drop	Toolbars and cascading menus	MS SmartTags	Right mouse button and forms	Center mouse button and pop-up menus
Annotation subject	Text	Text	File	Text	File
Automatic metadata extraction	Yes, using *Amilcare*	Yes, using *AeroDAML*	No	No	No

techniques into the desktop. Unfortunately, this only works for *Word*. Other editors would require similar enhancements.

Table 1 compares distinct approaches along the following dimensions:

- **coupling**, which denotes how semantic tooling (i.e. annotation, authoring) is coupled with either the resource format, the editor or the operating system. The editor alternative admits two additional options depending on whether the editor has been developed *ad hoc* for annotation purposes, or it is realized as a plug-in for an existing editor. According to this criterion, *seMouse* is the less coupled solution.
- **file-format support**, which indicates the type of formats the tool can handle. This is somehow related with the previous criterion in the sense that the lesser the coupling, the broader the file types handled by the tool. By working at the operating-system level, *seMouse* can be integrated with any editor available for *Windows*.
- **attachment availability**, which refers to the possibility of attaching the metadata to the file itself. Most formats permit to do so. This is an interesting option in case a file needs to be shared with other users or environments. However, the vocabulary and attaching mechanism used commonly depends on the format/editor used. For instance, Adobe is promoting the *Extensible Metadata Platform* (XMP) initiative[3] whereas *OpenOffice*[4] has a different set of metadata which also overlaps (but does not totally coincide) with *Dublin Core*[5]. To complicate things further, how metadata is attached to the document can also vary. As a result of this heterogeneity and proprietary formats, extracting the embedded metadata results in a high programming and maintenance effort as it has been particularized for each format.
- **GUI device for semantic interactions,** which refers to the GUI device used for annotation/authoring. Loose-coupling approaches complicate a seamless integration (if any) with the GUI of the chosen editor. For semantic-lite tooling, this can pose no problem as the mouse can give enough support. This is the option taken by

[3] http://www.adobe.com/products/xmp/main.html
[4] http://www.openoffice.org
[5] http://dublincore.org

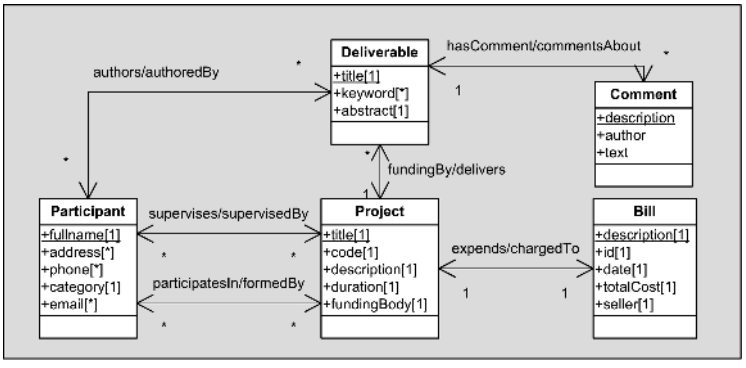

Fig. 1. A sample ontology

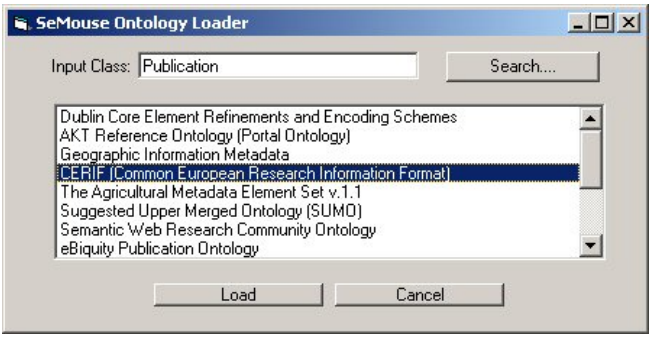

Fig. 2. SeMouse window to load the ontology

seMouse. However, more sophisticated tooling can make the mouse fall short, and require a tighter integration with the editor's GUI. *SemanticWord* is a nice example.

– **annotation subject**, which indicates the granularity of the element being annotated. Alternatives include the file as a whole, or regions of text within a file.
– **automatic metadata extraction,** where the availability of mechanisms for automatic annotation is indicated.

This work strives to introduce ontological concerns as seamless as possible into current desktops. Hence, our preferences align with those of *SemanticWord.* However, the myriad of formats which can be found in current desktop (e.g *.doc, .xml, .gif, .java, .pdf* to mention a few), and the corresponding editors, vindicate the use of an editor-independent solution. Our bet is to use the mouse to attain this aim.

Rather than using the extensibility technology provided by each editor (e.g. *ActiveX* controls in the case of *Word*), we move down to the operating system so that the solution can be available to no matter which editor. The result is *seMouse* (*S*emantic *MOUSE*), an annotation/authoring tool that achieves editor-independence by working at the operating-system level. By clicking on its middle button, data can be exported/imported from the *ontology* regardless of the editor you are working with.

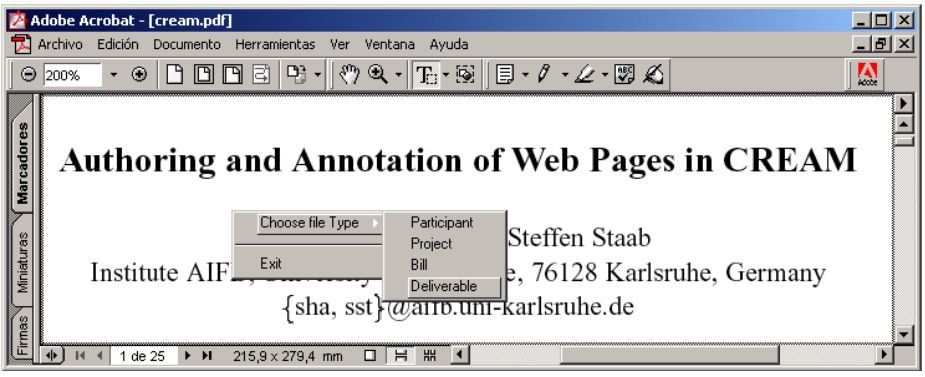

Fig. 3. Scenario 1: file classification

In this way, the user does not have to move to a new editor when annotating (like in *SMORE*), nor has to learn a new "ontological interface" when files from different formats are edited (like in *SemanticWord*).

The downside is usability. Enhancing current editors with "ontological concerns" certainly leads to more appealing and sophisticated interfaces. *SemanticWord* is a case in point. However, this advantage dilutes in a multi-editor scenario, where the *seMouse* approach ensures the same annotation/authoring tool no matter which editor is being used.

Another comparison of semantic annotation tools can be found in [14].

3 *seMouse* at Work

seMouse is an annotation/authoring extension of the mouse device that achieves editor-independence by working at the operating-system level. This section introduces *seMouse* with the help of an example.

Consider the heterogeneous documents that goes with a research project. This includes the project proposal (e.g. one Word file), bills payed with the project funding (e.g. twenty *Excel* files), papers as deliverables of the project (e.g. twenty files in both *.pdf* and *.doc* formats), participants (whose desktop counterpart can be either the homepage, an *.html* resource, or a *.pdf* resource) and comments (being realized as either emails or .doc resources).

Regardless of their format and folder location, it is likely that a high degree of content reuse as well as frequent contextual navigations within this "file space" happens. Being in a participant -an html file-, you frequently need to locate her project proposals -Word files-, or being in a project proposal, the associated papers -PDF files- are commonly accessed. This contextual navigation indicates the existence of a mental model. This mental model is made explicit through an ontology. This ontology serves then to configure *seMouse*. From then on, the mouse can be used to annotate & author file documents.

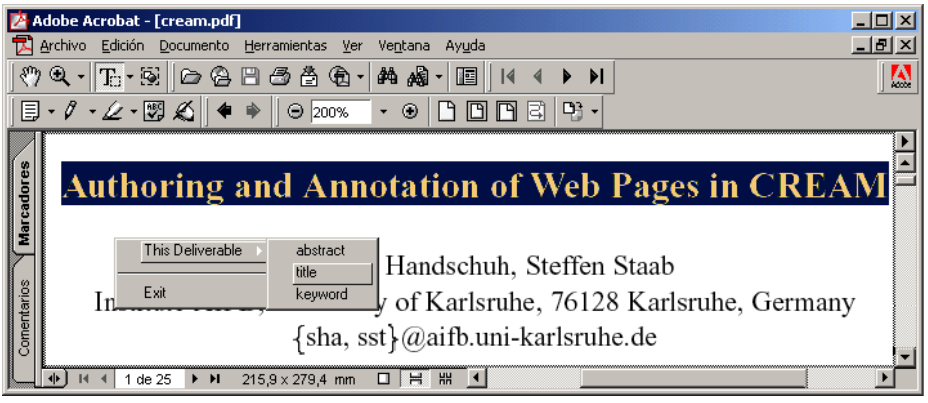

Fig. 4. Scenario 2: annotation. Some text is selected. Being a *deliverable* file, the menu displays properties of this class. The text will become the value of the chosen property.

As an example consider the ontology depicted in figure 1. It includes four classes which are characterized by a set of value-based properties (e.g. *title, keyword, abstract*). Associations are defined between these classes (e.g. a project is *supervisedBy* a participant).

However the tool is open to load any ontology stored in web ontology repositories[6] using the window shown in the figure 2 . The user introduces the name of a required class, and the system using the web services offered by the ontology repositories, shows in a scrollable listbox all the ontologies that contains the input class. The user must select the appropriate ontology clicking on the load button, and the ontology is uploaded on the semantic desktop.

Once seMouse is configured with the ontology, interactions with the underlying ontology are achieved via mouse clicks. Specifically, pressing the middle button on the mouse causes an interaction with the ontology manager (part of the seMouse installation). This interaction is context-schema sensitive, i.e. the button accomplishes distinct operations depending on the place the pointer sits on. Next paragraphs introduce five scenarios of the use of the semantic mouse.

Scenario 1: file classification (see figure 3). First of all, files need to be identified as instances of any of the ontology's classes. This is achieved by opening a file, and pressing the middle button. A menu pops up for the user to indicate to which class this file is a resource.

Scenario 2.1: property annotation (see figure 4). Annotation&authoring becomes the counterpart of copy&paste in traditional desktops, with the difference that now these operations are conducted along the ontology net. What is being exported(i.e. copy) is no longer a string but a class property of the ontology.

If a file has already been categorized, the annotation process may begin. If some text is selected, the mouse is used to export this text as part of the value of a property as it is shown in figure 4. Of course, the set of properties will depend on the class of

[6] schemaweb.info or swoogle.umbc.edu.

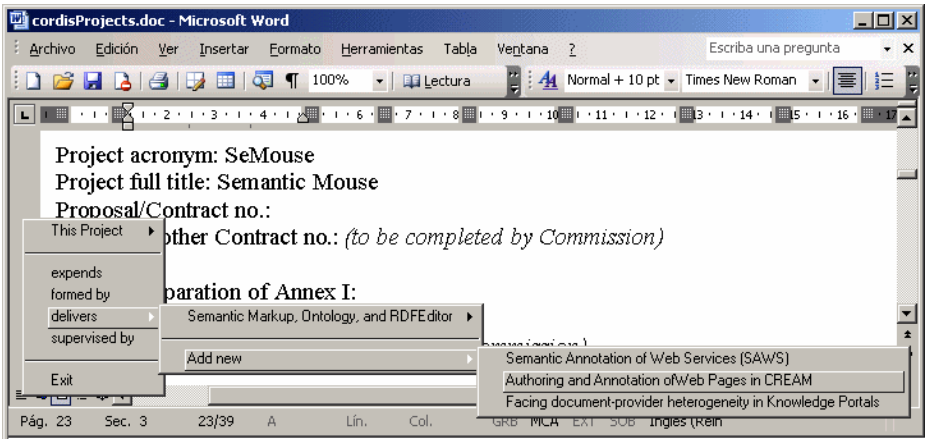

Fig. 5. Scenario 2: annotation. No text is selected. The menu shows associations of the file class.

the resource. In the example, *title, keyword* and *abstract* correspond to properties of the *deliverable* class.

Scenario 2.2: association annotation (see figure 5). Once a file has been categorized, if no text is selected, the middle button is used to establish associations with other files. This situation is exemplified in figure 5. In this case, the CORDIS project template [7] for EEC projects has been used. This Word document has been classified as *Project* class instance, and when the middle button is pressed, a menu pops up for the user to link the current resource with other target resources. The menu is customized for the current resource, that is, the associations are restricted to those available for the current resource, whereas the target of all the associations are also limited to those files of the appropriated class. In the example, if the user selects the *delivers* association, the association can only be established with *Deliverable* files, since this is the destination class of the *delivers* association. The upper part of the *delivers* menu shows all the associations already annotated.

Scenario 3: authoring (see figure 6). Associations being set during annotation can now be exploited. For instance, the *project* resource can import the title of its associated *deliverable* resources. In the example, the article *"Authoring and Annotation of Web Pages in CREAM"* appears as a *deliverable* of the current file. By selecting this article, the menu is extended right wise to show up its properties. The user can select one of these properties, and its value is inserted at the cursor place.

Scenario 4: semantic navigation (see figures 7 and 8). File location in current desktops frequently implies folder digging. By contrast, semantic navigation strives to exploit the associative behavior of the human memory. A resource can be located from the resources it is related to. That is, the ontology provides the context to facilitate resource location.

[7] http://dbs.cordis.lu/cordis-cgi/autoftp?FTP=/documents_r5/natdir0000035/s_2064005_20050316_104351_2064en.wd9.doc&ORFN=2064en.wd9.doc

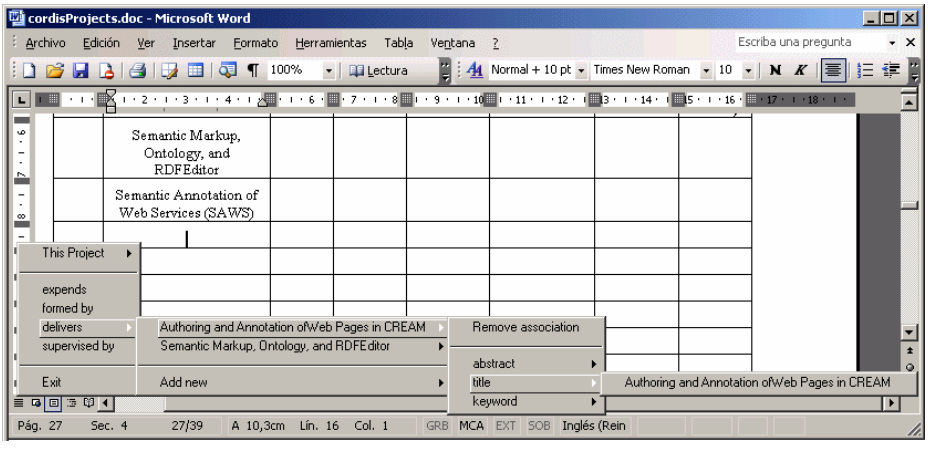

Fig. 6. Scenario 3: authoring. The *title* of a *deliverable* is imported into a *project* resource.

Once a file has been selected, semantically-related files can be located by pressing the middle button, regardless of the folders where these files are physically located, providing a resource-centric navigation. This facilitates location of neighbor resources, but it may be cumbersome whenever browsing is required. In this case, a graph-based RDF visualizer can be a better option (see [5] for an overview of RDF visualizers).

In this work, the *Welkin*[8] editor has been extended for our purposes. Figure 8 depicts the graph for our sample problem. Some of the nodes stand for resources (i.e. documents). *Welkin* has been extended so that clicking on one of these nodes makes the corresponding resource to be edited. In this way, *Welkin* becomes a "resource explorer".

If the user does not want to use external applications, another available way to navigate through the associations of the resource is shown in the figure 7. The user clicks the middle button on a resource, and selecting one of its associations, all the resources related with chosen association are displayed in a search window.

Scenario 5: ontology editing. Back to the desktop, the ontology can be edited by pressing the middle button of the mouse with no file selected. In this case, the associated action calls an ontology editor like Protégé [8].

Nevertheless, some authors argue that current tools for ontology creation are too difficult for ordinary users [15, 14]. The authors empirically found that *"the effort to select a class before typing in an annotation discouraged use of the tool"* (Protégé in this case)*"*. This observation is pertinent in the context of this work as we care for usability.

So far, *seMouse* is a tool for authoring and annotation, and we take the ontology for granted. However, the semantic desktop should provide for seamless ontology creation as well. In [15] the authors discuss an extreme approach to authoring whereby users immediately created metadata without defining the ontology first: *"it is our belief*

[8] http://simile.mit.edu/welkin/index.html

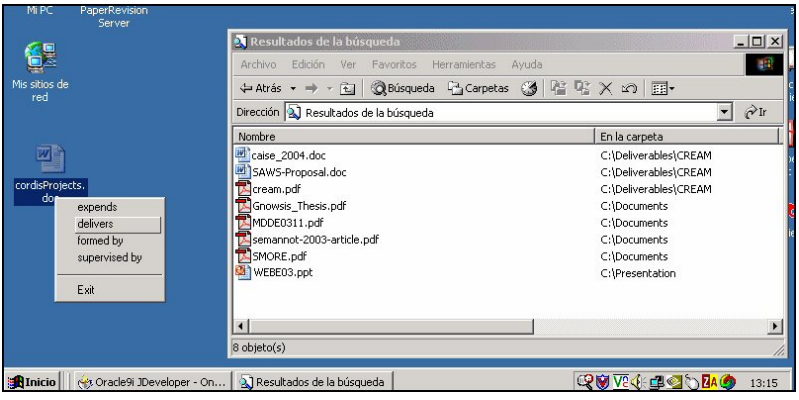

Fig. 7. Scenario 4: semantic navigation. The user can navigate along the associations: from a *project* resource to its deliverables.

that ontologies can be created later in a bottom-up fashion, as the by-product of creating and using data, rather than a straightjacket that inhibits the evolution of domain vocabularies".

We plan to extend *seMouse* for ontology creation. Rather than creating classes and properties out of the blue, *seMouse* will facilitate dynamic definition of classes and properties, as resources are being annotated and investigate on how much meta-data can be automatically inferred from the type and context of the resource.

4 Evaluating *seMouse* Usability

We adopt ISO's broad definition of usability [1] as consisting of three distinct aspects:

- *Effectiveness*, which is the accuracy and completeness with which users achieve certain goals. Indicators of effectiveness include quality of solution and error rates. In this study, we use quality of solution as the primary indicator of effectiveness, i.e. a measure of the outcome of the user's interaction with the system.
- *Efficiency*, which is the relation between (1) the accuracy and completeness with which users achieve certain goals and (2) the resources expended in achieving them. Indicators of efficiency include task completion time and learning time. In this study, we use task completion time as the primary indicator of efficiency.
- *Satisfaction*, which is the users' comfort with and positive attitudes toward the use of the system. Users' satisfaction can be measured by attitude rating scales such as SUMI [10]. In this study, we use preference as the primary indicator of satisfaction.

Subjects. The experiment was conducted among 6 PhD students. They have a good background on computing but they have never been exposed to semantic issues. Hence, a ten-minute talk was given introducing the purpose and functionality of *seMouse*.

Given material. Two documents were prepared. First, a UML diagram of the figure 1 describing the ontology and second, a document describing the set of 16 files of distinct

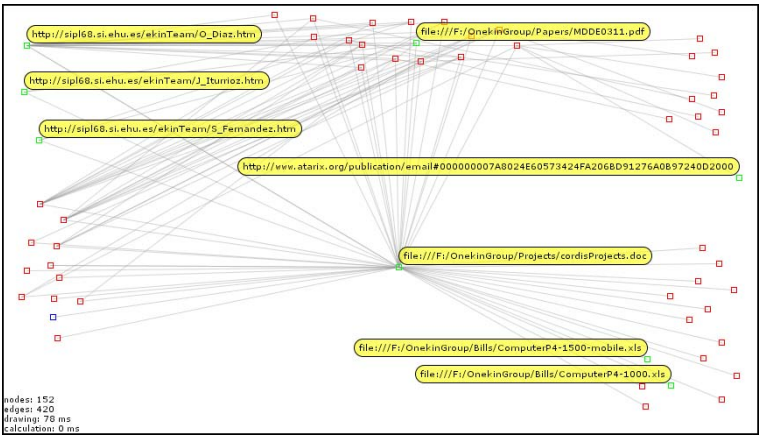

Fig. 8. Graphical representation of our sample domain using Welkin

types (PDF, DOC, PPT and HTML) and of different semantic concepts (Projects, Deliverables and Participants). Table 2 indicates for each file its class and its associations corresponding to a fictitious project. Students were familiar with the UML notation so they quickly caught the main classes and associations of the ontology.

Tasks. In the experiment each subject has to complete the task of creating a knowledge folder. This includes:

1. file classification. Each file should be classified according to its class (scenario 1). The 16 files correspond to 2 *projects*, 10 *deliverables*, and 4 *participants.*
2. annotation. The key property of each resource (i.e. *title, fullname,* etc) is annotated and the associations between the resources are now established (scenario 2).
3. authoring. Once a *project* document is open, take advantage of the relationships to fill in the *tables of participants* and *deliverables* as shown in scenario 3.

4.1 Results

Effectiveness. All the students complete their tasks without any additional help. This makes us think that the GUI is intuitive enough. It should be noted that the number of resources to be annotated, sixteen, was rather small, although there are not yet experiences on the average size of resources handled by a layman on his daily tasks. If the number of resources is too large, the solution is not within the scope of desktop tooling but content management frameworks. Nevertheless, the notion of knowledge folder also helps to split the resource bulk into meaningful clusters so that *seMouse* does not have to cope with too numerous resources.

Efficiency. The classification and “key attribute” annotation took on average, 30”. On the other hand, the students spent 10” on average to establish an association between

Table 2. Resources and their associations to be established by the participants during the test

	Key attribute	File type	Class	Relation
1	Semantic Web	.doc	Project	delivers:[3][4][5][6] [7][8][9][10] formedBy:[13][14][15]
2	Personal Information Management	.doc	Project	delivers:[3][4][5][6] [11][12] formedBy:[13][14][16]
3	Authoring and Annotation of Web Pages in CREAM	.pdf	Deliverable	fundingBy:[1][2]
4	Incremental Formalization of Document Annotations	.pdf	Deliverable	fundingBy:[1][2]
5	Trends in Database Development: XML, .NET, WinFS	.ppt	Deliverable	fundingBy:[1][2]
6	Semantic Annotation of Web Services (SAWS)	.doc	Deliverable	fundingBy:[1][2]
7	Semantic (Web) Technology in Action: Ontology Driven Information Systems for Search, Integration and Analysis	.doc	Deliverable	fundingBy:[1]
8	OWL: An Ontology Language for the Semantic Web	.ppt	Deliverable	fundingBy:[1]
9	The Semantic Desktop: an architecture to leverage document processing with metadata	.pdf	Deliverable	fundingBy:[1]
10	Towards the Self Annotating Web	.pdf	Deliverable	fundingBy:[1]
11	Mining the Semantic Web	.doc	Deliverable	fundingBy:[2]
12	Semantic Word Processing for Content Authors	.pdf	Deliverable	fundingBy:[2]
13	Steffen Staab	.html	Participant	participatesIn:[1][2]
14	Siegfried Handschuh	.html	Participant	participatesIn:[1][2]
15	Yolanda Gil	.html	Participant	participatesIn:[1]
16	Tim Berners-Lee	.html	Participant	participatesIn: [2]

Table 3. Questionnaire to assess *seMouse* usability

Questions about the method	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree	**User Results**
1. The annotation of documents enhances the navigation and localization of resources	❑	❑	❑	❑	❑	4.5
2. The time consumed in annotating a document, pays off	❑	❑	❑	❑	❑	4.5
General questions about SeMouse	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree	
3. The seMouse interface is well structured	❑	❑	❑	❑	❑	3
4. I understand the enabled operations at each time	❑	❑	❑	❑	❑	3.25
5. Overloading the mouse with the semantic operations is intuitive	❑	❑	❑	❑	❑	3.7
Questions about SeMouse functionality	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree	
6. The annotation of documents (class and properties) is intuitive	❑	❑	❑	❑	❑	4.7
7. The process of associating documents is intuitive	❑	❑	❑	❑	❑	3.85
8. The authoring of information is intuitive	❑	❑	❑	❑	❑	3.5

two resources. The last task, authoring, i.e. obtaining the *participants'* name and *deliverables'* title took on average, 15" per resource.

Satisfaction. To measure this property a questionnaire was provided to the subjects. Table 3 summarizes the result of the test along the Likert scale: Strongly agree (5), Agree (4), Neutral (3), Disagree (2), Strongly disagree (1). All the students agree on the usefulness of semantic annotation for improving location and navigation in the resource

space. As for *seMouse*, questions 3 and 4 show that the students were neutral about its GUI. More interesting is the deviations on the opinion about overloading the mouse with semantic operations: 2 strongly agree, 2 agree and 2 disagree. Although they all appreciate the format-independence provided by *seMouse*, some of them were not accustomed to use the middle button that they found counter-intuitive. This situation can however improve as the users get more practice on using this gadget. Finally, classification and annotation was found intuitive to be achieved through *seMouse*. Association establishment was found more complicated.

5 The *seMouse* Architecture

The DOGMA initiative [11] is promoting a formal-ontology engineering framework that basically consists of a three-layer architecture, namely, the layer of the heterogeneous data sources, the ontology layer, and the consumers' layer. Mappings between these layers are established with the help of wrappers that lift these data sources onto a common ontology model and of integration modules (mediators in the dynamic case) that reconcile the varying semantics of the different data sources.

Basically, we follow this architecture where both sources and consumers are restricted to be resource handlers (either readers or editors); the ontology model is OWL; and the flow between resource handlers and the ontology is achieved through the mouse. The issue of semantic heterogeneity is not addressed in this work.

The *seMouse* architecture comprises three components (see figure 9), namely, the ***OntologyManager*** component, which is realized using *Joseki* [9], the ***SemanticDesktop*** component, which is supported as a specialization of *WindowsXP,* and the ***ResourceHandler*** whose interfaces are supported by editors or readers of documents (see figure 9).

The *IOntologyManager* interface comprises methods to add/remove/update OWL triples as well as to query and check the existence of a given resource. All these methods find their realizations in the *Joseki* implementation where *OWL* triples are stored in the *Jena DBMS*[10].

The *resourceHandler* component holds two interfaces, *IReader* and *IEditor,* which permit to extract or paste data from/to a file, respectively. Readers such as Acrobat Reader only support *IReader*, whereas editors such as Word support both *IReader* and *IEditor.* Implementation wise, these interfaces are supported through the *WndProc* function of *Windows*. Whenever anything happens to a window, the operating system will call this function informing what has occurred. The *message* parameter contains the message sent. The resource handler can then trap any message. In this case, however, only two messages need to be caught: copy and paste.

The *ISemanticDesktop* interface describes those operations that mediate between the resource handlers and the ontology manager. Its functionality resembles a kind of clipboard, exporting and importing metadata among files (operations *getText()* and *setText()*). Interactions with the semantic desktop are accomplished by pressing the middle button of the mouse (e.g. the *sendMessage(WM_MBUTTONDOWN)* operation).

[9] http://www.joseki.org
[10] http://www.hpl.hp.com/semweb/jena.htm

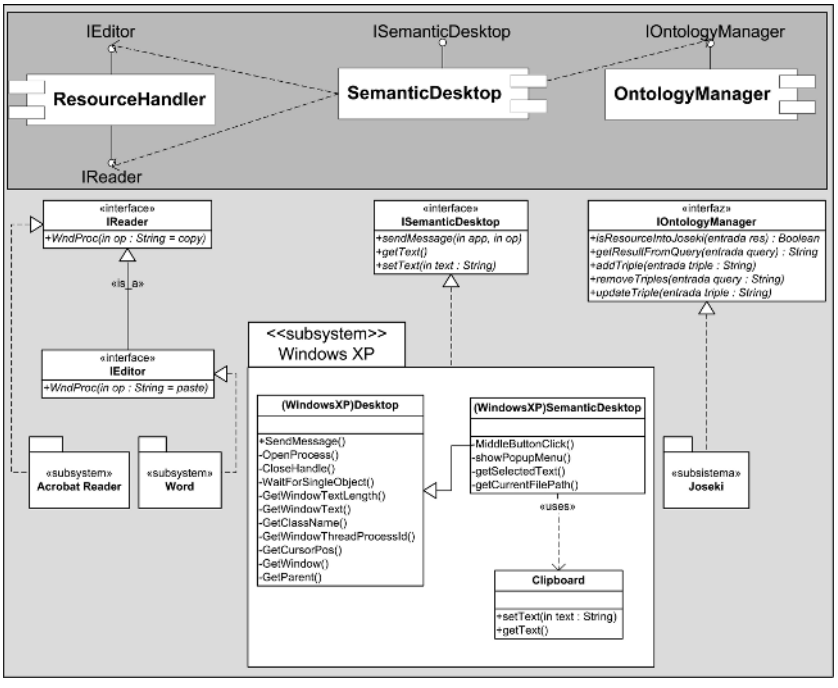

Fig. 9. The *seMouse* architecture

Figure 10 depicts the interaction diagram among the *seMouse* components. The *semanticDesktop* component has been unfolded to show its implementation classes, namely, *messageDispatcher, hook*[11]*, clipboard* and the *semanticDesktop* class itself.

Figure 10 considers an annotation scenario. A file is opened and some text is selected. By pressing the middle button of the mouse, a two step annotation process is initiated. First (1.1), the mouse device sends a middle-button click message to the *messageDispatcher*. When the *dispatcher* detects that there is a hook associated to this message, the *hook* is called (1.2); finally (1.3), the *hook* invokes the *semanticDesktop* which causes a menu to be popped up. In the second interaction (2.1), the user, through the mouse device, selects the annotation operation to apply to the selected text (2.2). The *semanticDesktop* sends a copy message to the *resourceHandler* through the *messageDispatcher*, and (2.3) the application copies the selected text into the clipboard. Finally (2.4), the text is retrieved by the *semanticDesktop* from the clipboard which, in turn, builds up the OWL triple.

[11] A Window *hook* is *"a point in the system message-handling mechanism where an application can install a subroutine to monitor the message traffic in the system and process certain types of messages before they reach the target window procedure"* [16]. So, hooks are basically event handlers that catch the message sent to the window. Through hooks, these messages can be modified or even discarded before they even reach the target window. In this implementation we caught just one message: *WM_MBUTTONDOWN* which is sent if the middle button on the mouse has been pressed. This has been attained using the *Cool Mouse* utility [7].

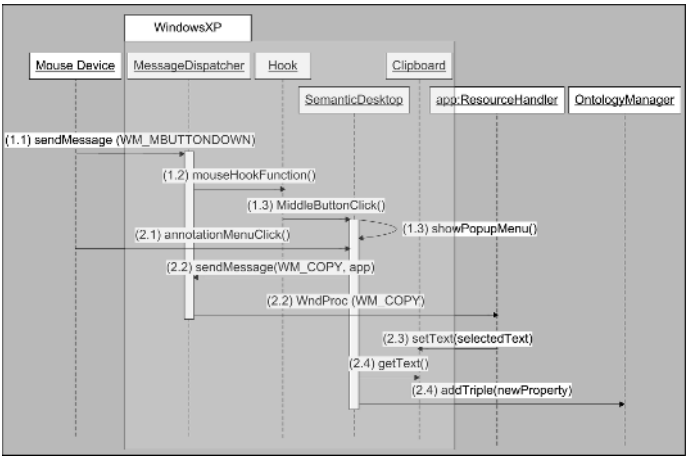

Fig. 10. Interaction diagram among the *seMouse* components. The *annotation* use case.

6 Conclusions

This work strives to lower the adoption barrier of the semantic desktop by providing seamless tooling. To this end, the mouse is proposed as the interactive device. In this way, traditional desktop operations are now re-interpreted and framed by the ontology: copy&paste becomes annotation&authoring, and folder digging becomes property traversal. Moreover, through the mouse, the user can classify, annotate, author, and locate a file as a resource of the underlying ontology. Being editor-independent, the mouse accounts for portability and maintainability to face the myriad of formats and editors which characterizes current desktops.

Similar to other areas of computing, a balance is needed between generality (e.g. format-independence, editor-independence, etc), and functionality (i.e. the semantic tooling available). *seMouse* illustrates a semantic-lite approach where a compact set of functions are available to no matter which editor within *Windows*.

Acknowledgments. This work is partially supported by the Spanish Science and Technology Ministry (MCYT) under contract TIC2002-01442. Sergio F. Anzuola enjoys a doctoral grant from the University of the Basque Country.

References

1. ISO 9241-11. Ergonomic requirements for office work with visual displays terminals(VDTs) - Part 11: Guidance on usability. Technical report, ISO, 1998.
2. Adam Cheyer, Jack Park, and Richard Giuli. IRIS: Integrate. Relate. Infer. Share. In *1st Workshop on The Semantic Desktop*, November 2005.
3. Stefan Decker and Martin Frank. The Social Semantic Desktop. Technical report, Digital Enterprise Research Institute (DERI), May 2004.

4. Benja Fallenstei. Fentwine: A navigational rdf browser and editor. In *Proceedings of 1st Workshop on Friend of a Friend, Social Networking and the Semantic Web*, August 2004.
5. John Gilbert and Mark H. Butler. Review of existing tools for working with schemas, metadata, and thesauri. Technical report, Hewlett Packard Laboratories, October 2003.
6. Siegfried Handschuh and Steffen Staab. Authoring and Annotation of Web Pages in CREAM. In *The Eleventh International World Wide Web Conference WWW2002*, pages 462–473, 2002.
7. Shelltoys Inc. Cool Mouse - Mouse Wheel and Middle Mouse Button Utility, 2004. http://www.shelltoys.com/mouse_software/index.html.
8. Stanford Medical Informatics. The Protégé; ontology editor and knowledge acquisition system, 2004. http://protege.stanford.edu/.
9. Jon Iturrioz, Oscar Díaz, Sergio F. Anzuola, and Iker Azpeitia. The Semantic Desktop: an architecture to leverage document processing with metadata. In *Proceedings of the VLDB Workshop on Multimedia and Data Document Engineering (MDDE'03)*, September 2003.
10. Kirakowski J. and Corbett M. SUMI: The software usability measurement inventory. *British Journal of Educational Technology*, 24(3):210–212, 1993.
11. Mustafa Jarrar and Robert Meersman. Formal Ontology Engineering in the DOGMA Approach. In *On the Move to Meaningful Internet Systems, 2002 - DOA/CoopIS/ODBASE 2002 Confederated International Conferences DOA, CoopIS and ODBASE 2002*, pages 1238–1254. Springer-Verlag, 2002.
12. Aditya Kalyanpur, James Hendler, Bijan Parsia, and Jennifer Golbeck. SMORE - Semantic Markup, Ontology, and RDF Editor. http://www.mindswap.org/papers/SMORE.pdf, 2004.
13. David R. Karger and Dennis Quan. Haystack: A User Interface for Creating, Browsing, and Organizing Arbitrary Semistructured Information. In *CHI 2004 Conference on Human Factors in Computing Systems*, April 2004.
14. Brian Kettler, James Starz, William Miller, and Peter Haglich. A Template-based Markup Tool for Semantic Web Content. In *4th International Semantic Web Conference 2005,ISWC2005*, November 2005.
15. Robert MacGregor, Sameer Maggon, and Baoshi Yan. MetaDesk: A Semantic Web Desktop Manager. In *Knowledge Markup and Semantic Annotation Workshop, ISWC 2004*, November 2004.
16. Steve McMahon. Win32 Hooks in VB - The vbAccelerator Hook Library, 2003. http://www.vbaccelerator.com/home/VB/Code/Libraries/Hooks/vbAccelerator_Hook_Library/article.asp.
17. The Gnowsis Project. Leo sauermann, 2005. http://www.gnowsis.com.
18. The Haystack Project. Haystack team, 2005. http:/haystack.lcs.mit.edu.
19. Marcelo Tallis. Semantic Word Processing for Content Authors. In *Workshop Notes of Knowledge Markup and Semantic Annotation Workshop (SEMANNOT 2003). Second International Conference on Knowledge Capture (K-CAP 2003)*, October 2003.