Semantic Clipboard - Semantically Enriched Data Exchange Between Desktop Applications

Gerald Reif, Martin Morger, and Harald Gall

University of Zurich, Department of Informatics, 8050 Zurich, Switzerland
{reif,gall}@ifi.unizh.ch
martinmorger@hispeed.ch,
WWW home page: http://seal.ifi.unizh.ch/

Abstract. The operating system clipboard is used to copy and paste data between applications even if the applications are from different vendors. Current clipboards only support the transfer of data or formatted data between applications. The semantics of the data, however, is lost in the transfer. The Semantic Web, on the other hand, provides a common framework that allows data to be shared across application boundaries while preserving the semantics of the data.
In this paper we introduce the concept of a Semantic Clipboard and present a prototype implementation that can be used to copy and paste RDF meta-data between desktop applications. The Semantic Clipboard is based on a flexible plugin architecture that enables the easy extension of the clipboard to new ontology vocabularies and target applications. Furthermore, we show how the Semantic Clipboard is used to copy and paste the meta-data from semantically annotated Web pages to a user's desktop application.

1 Introduction

Users frequently use the clipboard of the operating system to exchange data between desktop applications. They select the data they want to copy in the source application, the data is then temporarily stored by the clipboard, and inserted into the target application selected by the user. With current clipboard applications the data is transferred as text or binary data. The semantics of the data, however, is lost. As a consequence, the user has to restore the semantic context when pasting the data to the target application.

For example, a user orders a cinema ticket in an online-store and gets a Web page with the ticket's receipt. Using the operating system clipboard to add the event to the calendar or the cinema's address to the address book can not be done in one step. Instead, the user has to select each information item in an individual step (e.g., the movie title as event name, the street name, etc.) and paste it to the according field of the target application. This way the user manually restores the semantic context of the data.

To overcome this semantic gap, the Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries [22]. To make the semantics of the data machine-processable, the Semantic Web employes ontology-based meta-data that is formalized using the the Resource Description Framework (RDF) [13]. Therefore, Web pages in the Semantic Web have to be annotated with RDF meta-data that describe the semantics of the content.

Using a clipboard that is based on Semantic Web technologies enables the transfer from RDF meta-data between applications. This way a clipboard is able to preserve the data semantics during the data transfer. In the scenario described above, the clipboard can now paste the performance data in RDF format from the cinema's online-store receipt page to the user's desktop application. The desktop application interprets the well-defined semantics and performs the appropriate actions (e.g., creating a new event for the performance in the calendar, adding the cinemas address to the address book) without having the user to restore the semantic context of the data.

In this paper we discuss the semantic implications when pasting data to different desktop applications and introduce the architecture and a prototype implementation of a *Semantic Clipboard*. To enable the easy adaptation of the clipboard to new ontology vocabularies and target desktop applications, the Semantic Clipboard is based on a flexible plugin architecture. To demonstrate the potential of the Semantic Clipboard, we implemented plugins for several ontology vocabularies and desktop applications for the Apple MacOS and MS Windows operating systems.

The remainder of the paper is structured as follows. Section 2 discusses the idea that target applications define the semantic implications of the transferred data. Section 3 introduces the plugin architecture of the Semantic Clipboard and Section 4 presents the prototype implementation. Section 5 discusses related work and Section 6 presents future work and concludes the paper.

2 Semantic Implications Defined by Target Application

An ontology formally defines the semantics of the common terms that are used to describe the domain of discourse [5]. Since RDF meta-data is based on the terms defined in the ontology, machines have access to the semantics of the data. RDF data can be exchanged between applications without losing its semantics and the applications have the same semantic understanding of the data. The implications of the exchanged data, however, depend on the target application. The target application processes the data received and handles the data based on its semantic context. Therefore, pasting the data to different target applications can cause different semantic implications.

For example, the Web page of an online-banking application presents the user's latest bank account statement, shown in Figure 1. The semantic annotation of such a Web page contains information such as the date, the value, the recipient, and the posting details for each money transfer in RDF format. De-

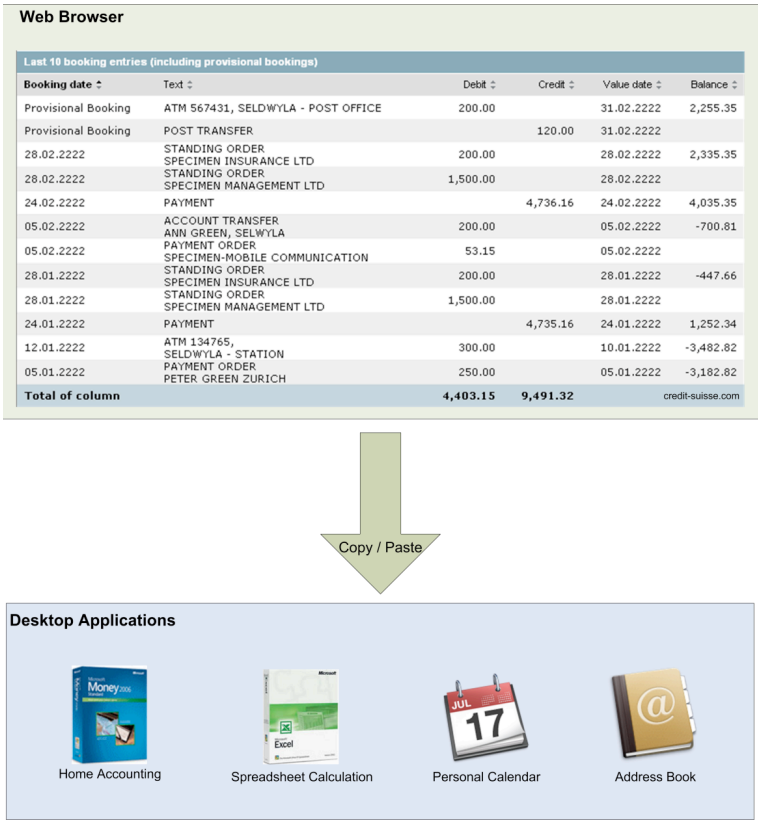

Fig. 1. The semantic implications are defined by the target application.

pending on the target application pasting the data will cause different actions, determined by the semantic context of the application.

The following list shows the different actions for some possible target applications:

- Home accounting application (such as MS Money) – A series of accounting records will be created; one for each money transfer.
- Spreadsheet applications (such as MS Excel) – The data will be formatted as a table; a new row is created for each posting line.
- Calendar application (such as iCal) – A new calendar entry will be created for each day a money transfer took place.
- Address book application (such as the MacOS Address Book) – The address of the recipients will be added to the address book.
- MP3 player (such as iTunes) – Since the semantic context of a MP3 player does not include money transfers, dates, or addresses the application is not able to perform any meaningful action with the data and will ignore it.

As shown in the examples above, using a Semantic Clipboard to paste the same RDF data to different target applications will cause different application specific actions. Therefore, the clipboard has to hand-over the RDF data to some application-specific code which processes the data in the semantic context of the application. Since the Semantic Clipboard should be easily extensible to new target applications, this application-specific code is hosted in application and operating system specific plugins. The plugin architecture of the Semantic Clipboard is introduced in the following section.

3 Architecture

The Semantic Web community defined numerous ontology vocabularies for various domains. Therefore, it is an important requirement that the Semantic Clipboard can easily be extended to new ontologies to facilitate these various domains. On the other hand, there are countless desktop applications that are potential target applications for a data transfer. Hence, it is a further requirement that the support for new desktop applications can easily be added to the Semantic Clipboard.

Both of requirements concern the extensibility of the Semantic Clipboard. To meet these requirements we propose to use a flexible plugin architecture for the Semantic Clipboard. This way the capabilities of the clipboard can be extended by adding new plugins and leaving the core of the implementation untouched. The basic components of the plugin architecture are shown in Figure 2.

The architecture is split into two main parts, the *core functionality* and the *plugins* that are used to increase the capabilities of the clipboard. The task of the core functionality is to load and manage the plugins, to provide the user interface, to parse the RDF data, and to manage the *clipboard sessions*. A clipboard session starts when the user requests the clipboard to read RDF meta-data. The data is then stored in the clipboard session and can be repeatedly pasted to the supported target applications. The current session ends and a new session is started when the user loads new RDF meta-data to the clipboard.

3.1 Reading from the RDF Source

The *Source Data Processor* is responsible for reading the RDF meta-data from the data source selected by the user. The Semantic Clipboard has to support several data sources. RDF data can be read from another desktop application that directly supports copying data in RDF format. RDF data can also be read form a file on the local hard-disk or from a source in the Internet via HTTP.

Semantically annotated Web pages are a further data source. These pages do not only provide their content in HTML format to be displayed by a Web browser but also RDF meta-data describing the semantics of the content. Several techniques exist to associate the HTML page and its RDF meta-data description [17]. When the RDF meta-data is embedded into the HTML page the Source Data Processor extracts the RDF data from the HTML page. In case the HTML

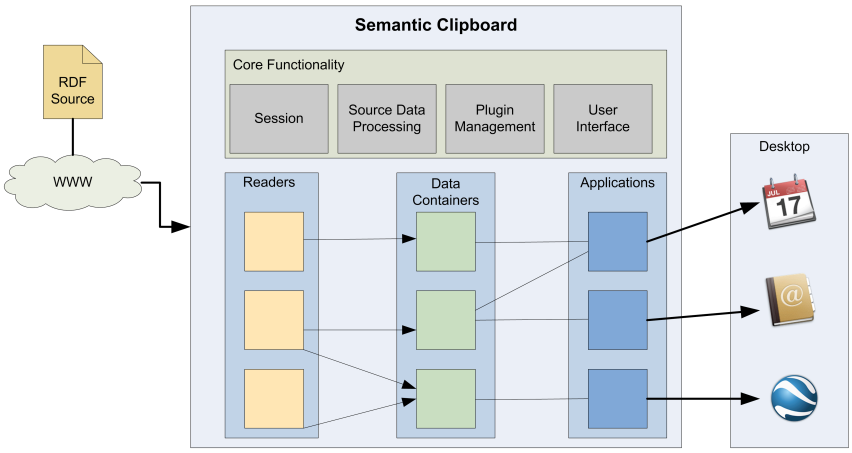

Fig. 2. Plugin architecture of the Semantic Clipboard. [15]

page contains a reference to an external document containing the RDF meta-data description, the Source Data Processor extracts the reference and retrieves its RDF meta-data over the network. Once the Source Data Processor loaded and parsed the RDF data, the data is handed over to the reader plugins, which are described in the following section.

3.2 Plugins

Using a plugin architecture allows developers to provide further plugins to enable the clipboard to handle new ontology vocabularies or to increase the number of supported target applications.

A possible architecture for the Semantic Clipboard would be to have only application-specific plugins. In this case the Source Data Processor directly hands over the RDF meta-data to the application plugin. This plugin is then responsible for analyzing the RDF data and for integrating the data into the semantic context of the application.

Such an architecture would have serious drawbacks. To support a new ontology, each application plugin that is able to handle data in the domain of the ontology has to be modified. In addition, application plugins that are able to process data based on the same ontology typically use the same statements to filter the RDF graph for specific information items such as the street name or the ZIP code of an address. Therefore, the same code would be found in several applications plugins.

To overcome these drawbacks, our proposed plugin architecture processes the RDF meta-data in three steps. The ontology-specific *reader plugins* extract the information from the RDF graph and store it in domain-specific *data container*

plugins. When the user selects the target application, the data containers are handed over to the responsible *application plugin* which communicates with the desktop application. The three processing steps are shown in Figure 2.

Introducing these additional steps increases the flexibility and code reuse in the architecture of the Semantic Clipboard. To support a new ontology in a domain where a data container already exists, only a new reader plugin for the new ontology has to be implemented. In addition, the code to extract the information from the RDF graph is only needed once, in the responsible reader plugin, even if several application plugins can handle the information encoded with this ontology.

In the following we introduce the plugin types in detail.

Reader plugin. A reader plugin is required for each supported ontology vocabulary (e.g., vCard [8], iCal [9], foaf [7]). The plugin extracts the information from the RDF graph that is encoded with the ontology vocabulary the plugin is responsible for. It then instantiates the corresponding data container plugin to store the extracted information.
A reader plugin can instantiate and write to several data containers. For example, the ontology provides the vocabulary to represent a person's contact information including its current geographical coordinates. The reader plugin would instantiate a contact container and a geographical container to store the data.

Data container plugin. The data container plugin is responsible for storing the information extracted by the reader plugins. A data container is responsible for a specific domain such as events or contacts. More than one reader plugin can write information to the same data container. For example, the RDF graph contains information about a person's contact encoded using the foaf [7] and vCard [8] ontology. In this case both the foaf reader and the vCard reader write its information to the same contact data container.
A data container can also store references to other data containers. For example, the RDF graph contains data about a meeting and its participants. The data container that stores the meeting event stores references to the contact data container of the meeting participants.

Application plugin. The application plugin builds the interface between the Semantic Clipboard and the desktop applications. When a user selects the target application, the responsible application plugin accesses the information stored in the data containers. The application-specific code in the plugin then communicates with the desktop application.
Depending on the semantic context of the desktop application, the application plugin can use information from several data containers. For example, the RDF graph contains data about a person including the current geographical coordinates and the data is stored in a contact and a geographic container. If the target application stores the geographic location in the person's contact entry, the plugin will access the information of both containers. On the other hand, if the semantic context of the target application does not overlap with the data in the current clipboard session no action can be issued in the target application.

Each application plugin has access to the user interface of the Semantic Clipboard to present its application-specific dialog. For example, the plugin asks the user to confirm the modification of an existing contact in the address book.

This flexible plugin architecture enables the Semantic Clipboard to be extended with regard to the number of ontology vocabularies the clipboard can handle (reader plugin), the domains the clipboard is aware of (data container plugin), and the target applications the information can be pasted to (application plugin).

3.3 Manifest Files

At compile time of the Semantic Clipboard, the number of plugins and their specific task is not known. To make the clipboard aware of the plugins that are available, each plugin has to provide a *manifest file* in a specific directory. The manifest of a plugin contains information about the plugin type, the associated class file, and the specific task the plugin was developed for. When the clipboard starts up, the *Plugin Manager* reads the manifest files and recognizes the plugins that are available, their tasks, and the dependencies between the plugins.

Beside the plugin name, the associated class file, and the plugin type (reader, data container, or application plugin) the manifest contains information that is specific to the plugin type. Since the Semantic Clipboard can evolve over time each manifest also contains a version number. The plugin-specific information in the manifest is listed below.

- **Reader plugin** - The manifest also provides the name and the version number of the data container(s) the extracted information is stored in.
- **Data container plugin** - No specific information is needed.
- **Application plugin** - The manifest contains name and version number of the data container(s) the plugin is able to handle. Since the application-specific code can depend on the operating system platform, the manifest contains the name of the operating system the plugin was designed for. Because the user of the Semantic Clipboard has to select the target application the data is pasted to, the manifest also contains the human-readable name and description of the target application, which is displayed to the user.

Figure 3 shows an example of the manifest file for the iCal application plugin. Lines 7 to 9 name the responsible class file, line 10 gives the plugin type, line 13 contains the operating system the plugin is implemented for, and lines 14 to 19 name the data containers the plugin is able to handle.

4 Implementation

In this section we present the prototype implementation of the Semantic Clipboard that is based on the plugin architecture as presented in the section above.

```
<rdf:RDF
   xmlns:semclip="http://seal.ifi.unizh.ch/semclip-rdf/"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
  <rdf:Description
   rdf:about="http://seal.ifi.unizh.ch/semclip-rdf/plugins#ICalAppPl
ugin">
    <semclip:classname>
      ch.unizh.ifi.seal.semclip.plugins.applications.ICalAppPlugin
    </semclip:classname>
    <semclip:plugintype>AppPlugin</semclip:plugintype>
    <semclip:appname>iCal Calendar</semclip:appname>
    <semclip:appdesc>The desktop calendar, redefined.</semclip:appdesc>
    <semclip:platform>Mac OS X</semclip:platform>
    <semclip:containers rdf:nodeID="A0"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A0">
    <semclip:containername>ICalContainer</semclip:containername>
    <semclip:containerversion>1</semclip:containerversion>
  </rdf:Description>
</rdf:RDF>
```

Fig. 3. Manifest file for the iCal application plugin.

The prototype is implemented in Java and includes plugins for several ontology vocabularies as well as desktop applications running on the Apple MacOS and the MS Windows operating systems.

Since the prototype of the Semantic Clipboard is not yet integrated in the clipboard of the operating system, the standard keyboard shortcuts `ctrl-c` and `ctrl-v` do not work. Instead we use a dialog window to select the RDF data source and the target application. Figure 4 shows the dialog window to select a local file or a document on the Web. The selected file is loaded by the Source Data Processor that checks whether the file is an RDF/XML document or an HTML Web page. If it is an HTML Web page the document is parsed and checked if an RDF fragment is embedded or if the page links to an external document with its RDF description.

The current implementation of the Semantic Clipboard supports embedded RDF in the HTML `<script>` element [17]. The used `application/rdf+xml` media type (MIMEtype) is defined in the RFC3870 [20]. An example of the use of the `<script>` element is shown below.

```
<head>
  <title>My Document</title>
  <script type="application/rdf+xml">
    <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
             xmlns:dc="http://purl.org/dc/elements/1.1/">
      <rdf:Description rdf:about="http://www.w3.org/"
                       dc:title="W3C␣Homepage"/>
    </rdf:RDF>
  </script>
  <body><!-- Web page --></body>
</head>
```

The HTML document is also searched for a reference to an external RDF description using the `<link>` element, which is the recommended technique to

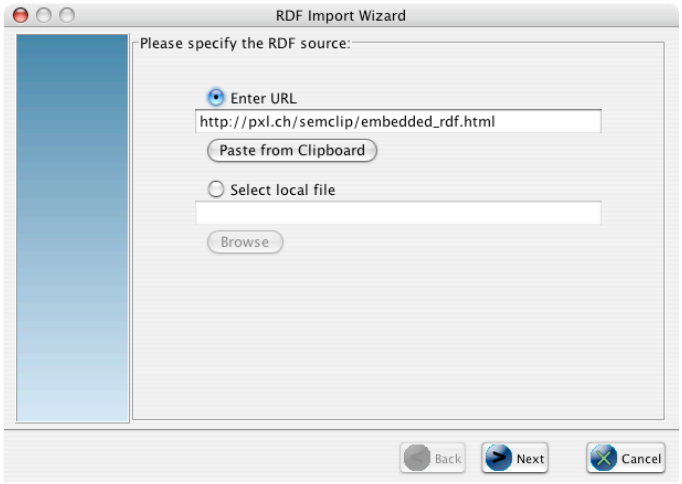

Fig. 4. Specifying the RDF data source.

associate the HTML Web page and its RDF description [19]. An example of the use of the `<link>` element is shown below.

```
<link rel="meta" type="application/rdf+xml" href="contact.rdf"/>
```

Once the RDF document is retrieved, it is parsed using the Jena RDF parser [11]. The Jena model is then handed over to the reader plugins. For the prototype we implemented readers for several ontologies in different domains. In the domain of contacts we implemented readers for the vCard [8] and the foaf [7] ontologies, in the domain of calendar events a reader for the iCal ontology [9] was implemented. In addition, we implemented a reader for a self-defined music ontology.

To store the information that is extracted from the RDF graph by the reader plugins, we implemented data containers for contacts, events, geographical coordinates, and music data.

In the last step the user is presented a dialog window with the list of applications the data can be pasted to. Since the manifest file of the application plugins includes the data containers the plugin is able to handle, the list only includes applications that are able to process the semantic context of the data in the current clipboard session. For example, if the RDF graph does not include calendar data, no event container is instantiated and, therefore, no calendar applications are listed. Figure 5 shows a screenshot of the application selection dialog of the Semantic Clipboard.

The application plugin is then responsible for the communication with the desktop application and for issuing the required actions. Depending on the operating system and the desktop application, the plugin uses different techniques to communicate with the desktop application. On MacOS the plugin can use Apple

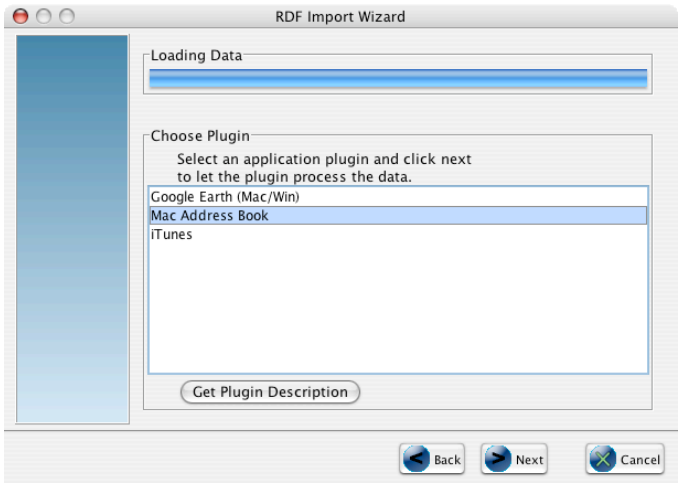

Fig. 5. Selecting the target application.

Script, whereas on MS Windows the plugin can use the Java COM Bridge JACOB [10] for the communication. A general technique, which works with most operating systems, is to write the data to a temporary file (e.g., an iCal file) and call the application with the filename as command-line parameter. Table 1 provides an overview of the application plugins we implemented for the prototype of the Semantic Clipboard as well as the data containers they can handle.

Data Container	**Target Application**		**Action performed**
	MacOS	**MS Windows**	
Contacts	MacOS Address Book	Windows Address Book	Contact is added
		Outlook	to the address book
Events	iCal	Outlook	New event is added
Coordinates	Google Earth	Google Earth	Focused on coordinates
Music	iTunes	—	Song is played

Table 1. Overview of the available application plugins.

Depending on the target application, some further user interaction might be necessary before the pasted information can be integrated in the semantic context of the application. When pasting contact data to a address book application, for example, the application requests the user to confirm the modification of an existing contact entry. Therefore, the application plugins have access to the Java Swing `JPanel` of the user interface of the Semantic Clipboard and can use it for the application-specific interaction.

5 Related Work

To our knowledge, the Semantic Clipboard we presented in this paper is the first implementation of a clipboard that is able to exchange semantically enriched data between desktop applications. In [2] Tim Berners-Lee first brought up the idea of a Semantic Clipboard. He proposed an architecture that uses conversation rules to convert RDF data from the ontology used by the application that provides the data to the ontology understood by the target application. In our implementation this conversation is implicitly done when a reader writes the extracted data to a data container and when an application plugin accesses this data.

The semiBlog [14] allows users to drag and drop data from desktop applications such as the Mac Address Book to the semiBlog editor. Wrappers take the data in the native data format of the application, convert it to RDF, and use it as semantic annotation of the blog entry.

The Live Clipboard [16] is a DHTML application that provides copy and paste functionality for data associated with a Web page. The data can be transfered within Web pages in XML format which does not address the data semantics. The Web Clipboard [23] is a semantic enabled extension of the Live Clipboard that is able to copy and paste RDF data. Instead of having to parse a complete resource description with each copy and paste operation, the Web Clipboard uses a small JSON snippet [12] which only contains an identifier of the resource and information where full information can be obtained from.

DBin [3] is general purpose Semantic Web application that enables users to create semantically enriched discussion groups where users can annotate any subject of interest. It provides a Semantic Clipboard to transfer data within DBin features such as queries, visualizers and exporters.

In addition, several approaches exist to benefit from Semantic Web technologies on the user's desktop [4]. In gnowsis [21] information from different applications is extracted, transferred from its native format into RDF meta-data, and stored in a central database. In this database information from different applications is interlinked which enables new views on the information that was scattered over several applications. In Fenfire [6], also meta-data from different sources is stored in a single RDF graph. The Haystack project [18] aims to provide users with a unified framework for managing their information, e.g., documents, emails, etc. through a single interface. Such a Semantic Desktop environment provides many RDF data source and target application the Semantic Clipboard can cooperate with.

6 Conclusion and Future Work

In this paper we introduced the concept of the Semantic Clipboard that enables data to be exchanged between desktop applications, without losing its semantics. We argued that two main requirements for such a clipboard are the possibility to add support for new ontology vocabularies and new target applications. To

meet these requirements we have developed a flexible plugin architecture that processes the RDF data in three steps. The ontology-specific reader plugin extracts the information from the RDF graph and stores it in domain-specific data container plugins. After the user selected the target application, the application plugin is responsible for the integration of the data from the containers into the semantic context of the desktop application.

The presented prototype implementation includes plugins for several ontology vocabularies, data containers, and desktop applications for the Apple MacOS and MS Windows operating systems. The prototype is able to process plain RDF/XML data and HTML Web pages with an associated RDF description. For the future we plan to add data that is encoded in RDFa format [1] as possible data source. RDFa is the W3C working draft for integrating RDF into XHMTL documents.

The prototype implementation of the Semantic Clipboard uses a different communication paradigm than the one that is found in current operating system clipboards. In an operating system clipboard the data source *pushes* the data into the temporal storage of the clipboard. The application, the user wants to paste the data to, *pulls* the data from this storage. The Semantic Clipboard, however, pulls the data from a file or the network and pushes it to the target application. We made this design decision not to establish this communication paradigm, but to be able to provide a proof of concept implementation with reasonable means. This way we did not have to modify the source applications to push RDF data to the clipboard and the target application to pull data from the clipboard. Regardless of the communication paradigm, the prototype implementation demonstrates the idea of pasting semantically enriched data between desktop applications. Nevertheless, a future version of the Semantic Clipboard should use the standard communication paradigm and be integrated in the operating system.

The prototype of the Semantic Clipboard can be downloaded under: `http://seal.ifi.unizh.ch/semclip`

References

1. B. Adida and M. B. eds. *RDFa Primer 1.0 - Embedding RDF in XHTML*, 16 May 2006. `http://www.w3.org/TR/xhtml-rdfa-primer/`.
2. T. Berners-Lee. Semantic Clipboard, January 2004. `http://www.w3.org/DesignIssues/SemanticClipboard`.
3. *DBin Porject Homepage*, Last visited October 2006. `http://dbin.org`.
4. S. Decker and M. Frank. The networked semantic desktop. In *Workshop on Application Design, Development and Implementation Issues in the Semantic Web at the 13th International World Wide Web Conference*, New York, USA, May 2004. CEUR Workshop Proceedings. `http://CEUR-WS.org/Vol-105/`.
5. J. H. ed. *OWL Web Ontology Language Use Cases and Requirements*. W3C Recommendation, 10 February 2004. `http://www.w3.org/TR/webont-req/`.
6. The fenfire project homepage, Last visited February 2005. `http://fenfire.org/`.
7. The friend of a friend (foaf) project homepage, Last visited February 2005. `http://www.foaf-project.org/`.

8. R. Iannella. Representing vCard objects in RDF/XML. W3C Note 22 February 2001, 2001. http://www.w3.org/TR/vcard-rdf.
9. iCalendar OWL ontology definition, April 7 2004. http://www.w3.org/2002/12/cal/ical.
10. Jacob - java com bridge, Last visited August 2006. http://sourceforge.net/projects/jacob-project/.
11. Jena - a semantic web framework for java, Last visited August 2006. http://jena.sourceforge.net/.
12. Introducing json (javascript object notation), Last visited August 2006. http://www.json.org/.
13. G. Klyne and J. J. C. eds. *Resource Description Framework (RDF): Concepts and Abstract Syntax*. W3C Recommendation, 10 February 2004. http://www.w3.org/TR/2004/REC-rdf-schema-20040210/.
14. K. Moller and S. Decker. Harvesting Desktop Data for Semantic Blogging. In *1st Workshop on the Semantic Desktop at the International Semantic Web Conference*, pages 79–91, Galway, Ireland, November 2006.
15. M. Morger. Semantic Clipboard. Master's thesis, Department of Informatics, University of Zurich, 2006.
16. R. Ozzie. Live clipboard technical introduction, Last visited August 2006. http://spaces.live.com/editorial/rayozzie/demo/liveclip/liveclipsample/techPreview.html.
17. S. B. Palmer. RDF in HTML: Approaches, June 2002. http://infomesh.net/2002/rdfinhtml/index.html.
18. D. Quan, D. Huynh, and D. Karger. Haystack: A Platform for Authoring End User Semantic Web Applications. In *International Semantic Web Conference*, pages 738–753, Florida, USA, October 2003.
19. W3C: RDF issue tracking: Issue faq-html-compliance: The suggested way of including RDF meta data in HTML is not compliant with HTML 4.01 or XHTML, January 2004. http://www.w3.org/2000/03/rdf-tracking/#faq-html-compliance.
20. RFC 3870: Application/rdf+xml media type registration. IETF RFC, September 2004. http://www.ietf.org/rfc/rfc3870.txt.
21. L. Sauermann and S. Schwarz. Gnowsis Adapter Framework: Treating Structured Data Sources as Virtual RDF Graphs. In *International Semantic Web Conference*, pages 1016–1028, Galway, Ireland, November 2005.
22. World Wide Web Consortium (W3C) Semantic Web activity homepage. http://w3c.org/sw.
23. Web clipboard demo, Last visited August 2006. http://www.sparqlets.org/clipboard/home.