Architectural Issues of the Semantic Clipboard as Ontology Mediation Service

Gian Marco Laube and Gerald Reif and Harald Gall

Software Evolution and Architecture Lab, University of Zurich,
Binzmuelestrasse 14, CH-8050 Zurich, Switzerland
gmlaube@access.unizh.ch
{reif, gall}@ifi.unizh.ch
http://seal.ifi.unizh.ch/

Abstract. When copying and pasting data between applications using the operating system clipboard, the semantics of the transfered information is usually lost. Using Semantic Web technologies these semantics can be explicitly defined in a machine process-able way. In previous research we developed a prototype to show the feasibility and benefits from a semantic enriched clipboard, that was limited to the number of ontologies it could handle or application that could access it. In this paper we introduce an advanced architecture for the Semantic Clipboard that incorporates the standard communication paradigm of operating system clipboards and is able to handle RDF graphs of arbitrary domains of interest. This architecture includes a data mediation service that overcomes vocabulary heterogeneities between source and target applications.

1 Introduction

The clipboard of the operating system is frequently used to transfer data between applications. The semantics of the data is, however, lost during the transfer. For example, if a user wants to copy an paste an address from a Web page to his address book application, he cannot mark the text block, representing the address, in the Web browser and paste it to the chosen application. Instead he has to copy and paste each individual information item (first name, last name, street, zip, *etc.*) to the corresponding field in the address book. This way, the user is responsible to preserve the semantic context of the information transfer.

The Semantic Web, on the other hand, aims to provide a common framework to enable applications to share information across application, enterprise and community boundaries [9]. In the Semantic Web, ontologies are used to formally capture the concepts and their relationships in a domain. RDF graphs then use the vocabulary defined in the ontology to describe items of the real world in a machine process-able way. Therefore, enabling the clipboard to transfer ontology based RDF graphs between desktop applications allows applications to share the same context.

In our previous research we developed a prototype of a Semantic Clipboard that can copy and paste data between desktop applications without losing its semantics [8]. In order to come up with a first prototype with reasonable means we took two drawbacks into account. (1) The clipboard does not follow the communication paradigm the users know from the operating system clipboard. In the prototype we pull data into the clipboard and push it to the target application. This paradigm we chose because it saved us from modifying the source and the target application. (2) The clipboard does not use RDF graphs to temporarily store the data to be transfered. Instead we use Java data containers, one specialized class for each domain. This way the clipboard has to be aware of the domain of the data to be transfered.

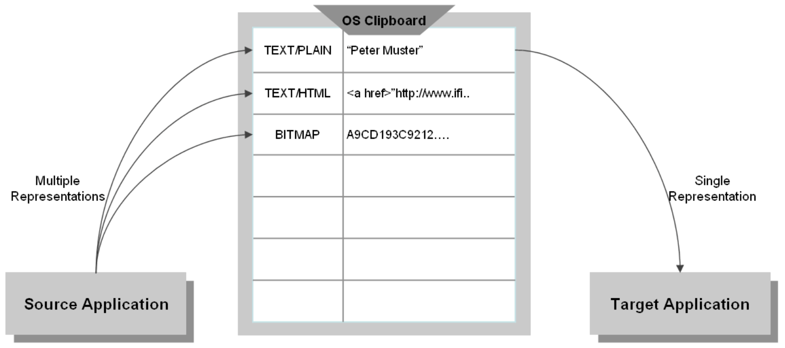
In this paper we present the architecture of a Semantic Clipboard that aims to overcome the shortcomings of the current implementation and is independent from client applications and ontologies. The Semantic Clipboard follows the common communication paradigm of operating system clipboards and can therefore be seamlessly integrated into them. Offering divers standardized interfaces it is simple to connect existing applications with the Semantic Clipboard. The architecture foresees an ontology mapping service which is able to mediate between the data offered by the source application and the ontology vocabulary requested by the target application.

The remainder of the paper is structured as follows. Section 2 discusses the working style of operating system clipboards. Section 3 introduces the revised architecture of the Semantic Clipboard. This proposal is illustrated in more detail in Section 4 by using the scenario from this introduction. Open issues of the current architecture are discussed in Section 5. Our architecture is compared to related work in Section 6 and a conclusion is given in Section 7.

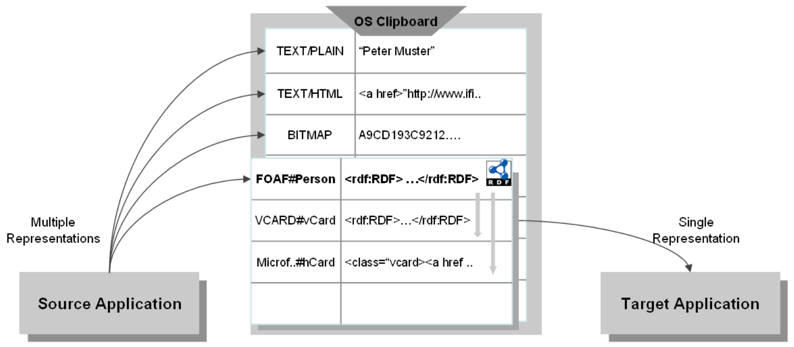
2 The Operating System Clipboard

Since it should be possible to integrate the Semantic Clipboard into the clipboard of the operating system, we briefly discuss its functionality. A source application is not only able to push plain text to the temporary storage of the clipboard, but also some alternative representations of the same data. Often MIME types [3] are used to identify the different versions of the data. For example in Figure 1(a) the source application pushed the data as plain text (`text/plain`), as HMTL (`text/html`), and as JPEG graphic (`image/jpeg`) to the clipboard. The target application has then the possibility to check whether the data is available in the preferred MIME type and request it from the clipboard. If the chosen representation is not available it can still pick the plain text version as the least common denominator.

As shown in Figure 1(b) the architecture of the Semantic Clipboard takes up the principle described above and extends it with new representation types and some additional value-adding features. Using the interaction paradigm of operating system clipboards the Semantic Clipboard could be directly integrated with them. This architecture and its mapping and querying service will be introduced in the next section.



(a) Default interaction with OS clipboard



(b) Semantics as additional representations and OS clipboard as data mediator

Fig. 1. A semantic enriched OS clipboard. Adapted from the MSDN Library description of the Windows Clipboard: `http://msdn2.microsoft.com/en-us/library/ms997518.aspx`

3 Architecture

In this section we introduce the architecture of the Semantic Clipboard and discuss the involved components. The architecture is depicted in Figure 2.

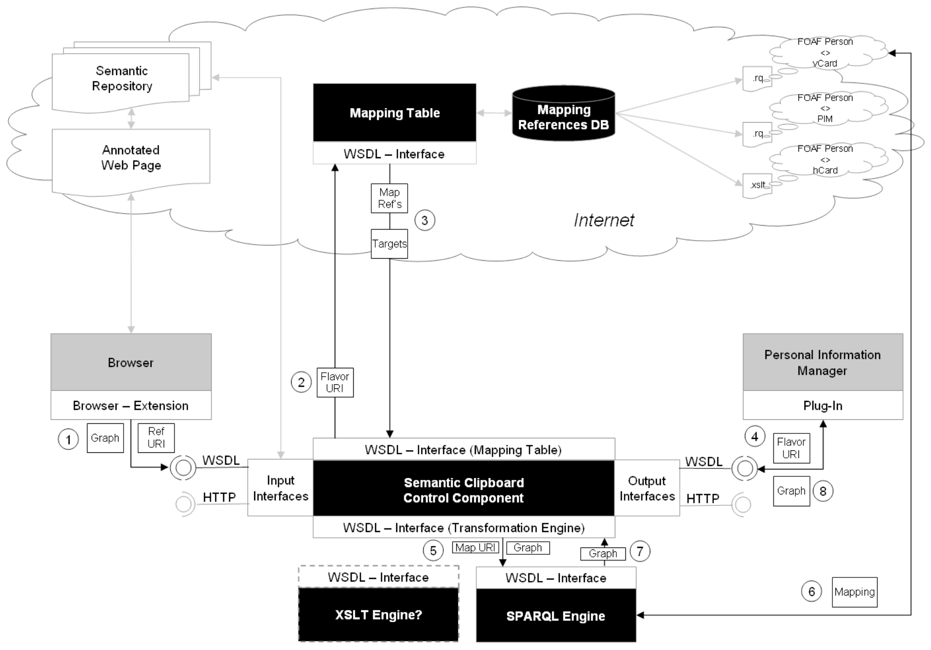


Fig. 2. The Architecture of the Semantic Clipboard

Control Component The Semantic Clipboard consists of three independent components, but there is one core Control Component that manages the actual storage and handling of all RDF graphs received. It is invoked every time an application pushes an RDF graph onto the Semantic Clipboard. The Control Component manages the different representations and possible mappings and answers data retrieval requests. The Control Component resides locally on the user's desktop.

Mapping Table The Mapping Table is an online Web Service and can be accessed by Semantic Clipboards or other applications. It manages and stores references to mappings between different ontologies that have an overlapping domain. Following a couple of guidelines, everyone can define such mappings and register them on the Mapping Table, where the references to them can be requested from personal Semantic Clipboards.

Transformation Engine Like the Control Component, the Transformation Engines are local components. The Transformation Engine is responsible to transform the the stored RDF graph into the format requested by the target application. Multiple Transformation Engines can be installed. A SPARQL [7] engine is used to process RDF graphs. Another implementation of a Transformation Engine could be based on XSLT [2] to deal with general XML graphs or other data formats. As mappings are already predefined for the use in a particular Transformation Engine these engines have a very simple dynamic model. After a target application requested a certain data representation the Control Component forwards the graph and a mapping reference to the Transformation Engine. The Engine downloads and executes the mapping on the graph and returns the result.

3.1 The Functions of the Semantic Clipboard

Store and Retrieve Data Graphs Any source application can send a data graph to the Semantic Clipboard and any other target application can later retrieve it. Independent of any additional services of the Semantic Clipboard described below, the source and target applications can always exchange arbitrary serialized data items over the Semantic Clipboard in its original format or ontology.

Handling of Different Representations A source application has the possibility to transmit several representations of one graph to the clipboard. Similar to the interaction with the Windows clipboard the target application can request a certain representation, check the availability of one specific or get an enumeration of all available types. Because of the variety of different ontologies, MIME types that are used by the Windows clipboard, are not ideal for the Semantic Clipboard. Rather general URIs would be suitable. But it is still open for discussion, if the Semantic Clipboard should manage representations on an ontology level (*e.g.,* FOAF graph) or on a class level (FOAF:Person graph). More on this subject can be found in section 5. For both options it is important that namespace declarations are transmitted with the data. The RDF graphs that are copied and pasted by the user are often only snippets of the original document. Such RDF snippets allow the user to select the information more precisely, as when copying complete documents. But the Semantic Clipboard can not distinguish between different representations nor provide any query or mapping services, if it can not refer to the type of the content it has received.

Query Service or Completion of the Graph Some semantically annotated Web pages do not provide the full description of a resource but only give references to other resources. In the example of Listing 1.1 only the information is given that a person with the id `GianMarcoLaube` knows a person identified as `GeraldReif`. If this data is pasted to an application, the application does not know the name of either one of those persons. This information is only available at the detailed description of the corresponding instance within the Web application. Therefore we introduce the Query Service in the architecture to request this information on demand. To be able to issue such a query, the Web application not only has to provide the semantic meta-data but also a hint where to find the detailed information for the resources (*e.g.,* a link to the knowledge base of the Web application).

Listing 1.1. Annotation of Web page without details

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
 xmlns:foaf="http://xmlns.com/foaf/0.1/">
    <foaf:Person rdf:about=
        "http://www.seal.uzh.ch/staff#GianMarcoLaube"/>
        <foaf:knows>
            <foaf:Person rdf:resource=
                "http://www.seal.uzh.ch/staff#GeraldReif"/>
        </foaf:knows>
    </foaf:Person>
</rdf:RDF>
```

Mapping Service Providing mappings between different representations is a core functionality of the Semantic Clipboard. We believe that most applications are bound to one specific ontology and can not translate its data into other representations. Also our architecture is more flexible regarding the natural evolution of these ontologies. To be able to provide mappings, the Semantic Clipboard first has to identify the content of the graph. Again the question arises, if ontologies or the actual classes are used to define the type of a graph. Independent of it, the representation can always be identified by one or multiple URIs. These URIs are checked with the Mapping Table to retrieve references of mappings that could transform the received representation in another type. To avoid unnecessary calculation time the mappings are executed "lazy", meaning on demand of the target application. The actual transformations are delegated to the Transformation Engines, which are also responsible for downloading and caching the mappings.

By defining the Mapping Table as Web Service everyone can share mappings with the community. For a broad adaption of this concept in the future, additional trust mechanisms would be needed.

4 Scenario for the Semantic Clipboard

The detailed interaction between the Semantic Clipboard modules can be best illustrated by defining a simple scenario. In the following example, a user selects and copies a contact address defined in the FOAF ontology [1] within his Web browser and pastes it into his personal information manager (*e.g.,* Microsoft Outlook) which is able to handle RDF graphs using the vCard ontology vocabulary [4].

Pushing Data on the Semantic Clipboard An extension of a browser extracts an RDF annotation from a Web page that the user has selected to be copied. The Control Component of the Semantic Clipboard receives the graph (Listing 1.2) trough his WSDL interface (Step 1 in Figure 2).

Listing 1.2. Example RDF Annotation transmitted to the Semantic Clipboard

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:foaf="http://xmlns.com/foaf/0.1/">
    <foaf:Person rdf:about=
        "http://www.seal.ifi.uzh.ch/students#GianMarcoLaube">
        <foaf:name>Gian Marco Laube</foaf:name>
        <foaf:firstName>Gian Marco</foaf:firstName>
        <foaf:surname>Laube</foaf:surname>
        <foaf:nick>gm</foaf:nick>
        <foaf:mbox>gmlaube@access.uzh.ch</foaf:mbox >
        <foaf:homepage rdf:resource="http://seal.ifi.uzh.ch" />
        <foaf:schoolHomepage rdf:resource="http://www.uzh.ch/"/>
        <foaf:knows>
            <foaf:Person rdf:resource=
                "http://seal.ifi.uzh.ch/staff#GeraldReif"/>
        </foaf:knows>
    </foaf:Person>
</rdf:RDF>
```

Besides the graph the source application can send other optional parameters to the Semantic Clipboard. In this case we assume the browser extension forwards also a reference URI (http://www.seal.uzh.ch/staff#) of a repository containing additional information about the content of the graph. But no further details about the type of the graph (class URI) or the element of interest (subject URI) are given.

Completion and Identification The example graph contains three *unresolved resource references*. Unresolved resources are resources that are used as object and further described by other triples (not used as subject of another triple within this graph). While the analysis of the first two resources (`http://seal.ifi.uzh.ch` and `http://www.uzh.ch`) by the Query Service does not result in additional information, more statements can be retrieved for the resource about Gerald Reif (`http://seal.ifi.uzh.ch/staff#GeraldReif`).

Listing 1.3. Additional Statements found by the Semantic Clipboard

```
<foaf:Person rdf:about=
        "http://seal.ifi.uzh.ch/staff#GeraldReif">
        <foaf:name>Gerald Reif</foaf:name>
        <foaf:firstName>Gerald</foaf:firstName>
        <foaf:surname>Reif</foaf:surname>
        <foaf:nick>geri</foaf:nick>
        <foaf:homepage rdf:resource="http://seal.ifi.uzh.ch" />
        <foaf:schoolHomepage rdf:resource="http://www.uzh.ch/"/>
        <foaf:knows>
                <foaf:Person rdf:resource=
                        "http://seal.ifi.uzh.ch/students#GianMarcoLaube"/>
        </foaf:knows>
        <foaf:knows>
                <foaf:Person rdf:resource=
                        "http://seal.ifi.uzh.ch/staff#HaraldGall"/>
        </foaf:knows>
</foaf:Person>
```

To find mappings for the graph we need to identify its content. For our scenario we assume the Semantic Clipboards identifies this graph by the class of the root element (rdf:type foaf:Person).

Check for Mappings The Clipboard Control takes all identified type URIs and forwards them as a Web Services Request to the Mapping Table (Step 2 in Figure 2). Every tuple (see Table 1) in the database of the Mapping Table contains a source and a target URI together with a reference to the mapping between those concepts.

Property	**Value**
Source URI	http://xmlns.com/foaf/0.1/Person
Target URI	http://www.w3.org/2006/vcard/ns#VCard
Mapping Reference	http://www.seal.uzh.ch/mappings/foafPerson_vCard.rq
Mapping Type	SPARQL

Table 1. Example of a Mapping Table Entry

The Mapping Table returns the of targets and mapping references (both URIs) together with the mapping type to the Clipboard Control (Step 3, Figure 2). The mapping type is used to choose the correct Transformation Engine when the respective representation is requested. Additionally to the shown example, other mappings could include *e.g.,* transformations to the PIM ontology or a XSLT script to generate a hCard.

Requesting a Graph Our target application, *e.g.,* a plug-in for Microsoft Outlook receives a paste command from the user and checks the Semantic Clipboard for its content. Lets assume, this plug-in can import contacts only as vCards and directly requests this representation using the WSDL interface of the Semantic Clipboard (Step 4, Figure 2).

The Semantic Clipboard checks its temporal storage. The get function operates as proxy method and initializes the transformation of the FOAF graph into a vCard, if the graph of this type is not already present. The Control Component forwards the mapping references and the graph to the SPARQL engine (Step 5, Figure 2). Before downloading the mapping, the Engine checks its cache for the mapping to avoid unnecessary downloads (Step 6, Figure 2). The SPARQL construct queries used are somewhat cumbersome to define manually (see listing 1.4), but it should be fairly easy to write semi-automated generators for them. These generators could enable developers to define mappings by simply drawing links between concepts of two different ontologies. SPARQL is also powerful enough to allow for more complex mappings than those that are needed in our scenario.

Listing 1.4. Extract of the SPARQL construct query from FOAF to vCard

```
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
PREFIX vCard:<http://www.w3.org/2006/vcard/ns#>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
CONSTRUCT {
        ?x rdf:type vCard:VCard .
        ?x vCard:Name ?name .
        ?x vCard:title ?title .
        ?x vCard:given-name ?firstName .
        ?x vCard:family-name ?familyName .
        ?x vCard:nickname ?nick .
        ?x vCard:email ?email .
        ?x vCard:url ?homepage .
        {...}
}
WHERE {
        ?x rdf:type foaf:Person .
                OPTIONAL { ?x foaf:name ?name . }
                OPTIONAL { ?x foaf:title ?title . }
                OPTIONAL { ?x foaf:firstName ?firstName . }
                OPTIONAL { ?x foaf:surname ?familyName . }
                OPTIONAL { ?x foaf:nick ?nick . }
                OPTIONAL { ?x foaf:mbox ?email . }
                OPTIONAL { ?x foaf:homepage ?homepage . }
                {...}
}
```

The translated graph is then returned to the Clipboard Control (Step 7, Figure 2) and to the Outlook plug-in that requested it (Step 8, Figure 2). In our example the resulting graph will contain two vCards. It is the responsibility of the Outlook plug-in to ask the user if both or only one of those should be imported into the address book. The final graph (Listing 1.5)is not as pretty as if defined manually, but fulfills its purpose:

Listing 1.5. Resulting vCards after the Transformation

```
<rdf:RDF
        xmlns:foaf="http://xmlns.com/foaf/0.1/"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:vCard="http://www.w3.org/2001/vcard-rdf/3.0#" >
        <rdf:Description rdf:nodeID=
                "http://seal.ifi.uzh.ch/students#GianMarcoLaube">
                <vCard:family-name>Laube</vCard:family-name>
                <vCard:url rdf:resource="http://seal.ifi.uzh.ch"/>
                <vCard:nickname>gm</vCard:nickname>
                <vCard:given-name>Gian Marco</vCard:given-name>
                <rdf:type rdf:resource=
                 "http://www.w3.org/2001/vcard-rdf/3.0#VCard"/>
                <vCard:email>gmlaube@access.uzh.ch</vCard:email>
                <vCard:Name>Gian Marco Laube</vCard:Name>
        </rdf:Description>
        <rdf:Description rdf:nodeID=
                "http://seal.ifi.uzh.ch/staff#GeraldReif">
                <vCard:family-name>Reif</vCard:family-name>
                <vCard:url rdf:resource="http://seal.ifi.uzh.ch"/>
                <vCard:nickname>geri</vCard:nickname>
                <vCard:given-name>Gerald</vCard:given-name>
                <rdf:type rdf:resource=
                 "http://www.w3.org/2001/vcard-rdf/3.0#VCard"/>
                <vCard:email>reif@ifi.uzh.ch</vCard:email>
                <vCard:Name>Gerald Reif</vCard:Name>
        </rdf:Description>
</rdf:RDF>
```

5 Open Issues in the Architecture of the Semantic Clipboard

5.1 Level of Detail in Content Identification

The Semantic Clipboard needs to be able to communicate about the type of graph that it stores for three reasons: (1) To distinguish between different representations. (2) To negotiate with target applications about their needs. (3) To search for mappings that it could provide on the graph. URIs can be used as identifier for the type of a graph. Two issues have to be discussed regarding these identifiers.

(1) There are "semantic" data formats that do not use URIs as identifiers and are not defined in a namespace. A prominent examples for these are Microformats. While one parser can be used to identify all Semantic Web graphs, Microformats or general XML graphs need special algorithms. Alternatively, source application could add the type as parameter when it copies the data on the Semantic Clipboard. Still, the Semantic Clipboard needs a URI as identifier for each of these formats, to be able to communicate about it.

(2) The second and by far more complex question is on what detail level applications should negotiate about the different representations or types of the graph. There are at least three options when identifying Semantic Web content:

– Ontologies
– Classes
– Graph patterns

Our architecture primarily aims at the first two options and we're currently evaluating the advantages and disadvantages of them. While the use of general ontologies avoids complex analysis of the graphs, mappings can be defined more flexible and in more detail when talking about the classes that they contain.

5.2 Further Evaluation of the Prototype Implementation

While the target application can always choose to consume the original graph, the chances are likely that it will make use of a representation generated by the Semantic Clipboard. The benefits and costs of providing such additional services like mapping and querying will have to be evaluated, when a mature version of the prototype has been implemented.

Another issue to be analyzed will be the complexity of defining or generating mappings for the Semantic Clipboard. Subclass relationships and mappings are a special field of interest. Either graphs could be analyzed with reasoners at run-time to find matching mappings or multiple transformations are generated when defining the mappings.

The quality of transformed graph will also need to be analyzed. There will be certainly some inconsistencies and other problems that need special attention or new rules. Guidelines will have to be written, that govern the definition of new mappings to avoid common mistakes.

We doubt that our architecture will be a very efficient solution, but that was never a design principle we were looking for. It will be more important to check the feasibility of our approach to look up mappings online and execute transformation between copy and paste operations.

6 Related Work

The main characteristics and drawbacks of the OS Clipboard were already presented in Section 1. There exist at least two initiatives that have a similar scope like the Semantic Clipboard.

The Live Clipboard [6] has received much attention from developer blogs. It's a DHTML application to move data from one Web site to another, or between the Web and standard applications. The Live Clipboard uses its own XML format as a wrapper around the data to be transmitted (mostly Microformats like hCard or hCal). The trick is to transfer this wrapped data as plain text representation in the normal OS clipboards. Data is thereby passed by value and and multiple representations can be defined within the wrapper sequentially. The Live Clipboard was already partially integrated with the most well known Browsers, but is not supported in their newest version. Screencasts show already the features the user can expect in the near future. An example includes dragging and dropping contacts from and into Web forms.

By transferring the data as plain text using the OS clipboard, the Live Clipboard inhibits the user from copying semantic and non-semantic data in parallel. Also applications need a special parser to interpret the Live Clipboard graphs. In contrast the Semantic Clipboard provides interfaces that accept and transmit RDF graphs in their original form.

The Web Clipboard [5] is a extension of the Live Clipboard. It also transfers meta-data using the plain text container of OS Clipboards. Instead of copying and pasting a complete resource like the Live Clipboard, it passes data by reference and forwards a JSON snippet[1] that includes only an identifier of the resource, information about a service endpoint and its capabilities. The client of the Web Clipboard can then use this meta-data to retrieve the information using the protocol and mechanism of its choice. A SPARQL interface is one possible capability that a data source can provide as interface. This interaction style, possible trough the use of global unique URIs, is certainly an improvement to the classical clipboards implemented in common operating systems. But not to define a standard protocol for the retrieval of the data mentioned in these JSON references, could also be an obstacle for the Web Clipboard. Both Web and Live Clipboard allow also to submit RSS streams and therefore provide "live updates" of data that has been transferred earlier trough the web, a particularly stunning effect for users new to such web technologies.

While the Live Clipboard transfers data by value and the Web Clipboard by reference, the Semantic Clipboard aims at providing both options, similar

[1] http://json.org/

to what current OS Clipboard offer. Applications that connect to it don't need to write special parsers, as it would be necessary to generate or interpret Live Clipboard data, but can call operations similar to those of the OS Clipboard to set or request the content of the Semantic Clipboard.

7 Conclusion

This paper shows how OS clipboards can be extended with Semantic Web technologies. Copying and pasting RDF graphs allow application to transfer data between each other without loosing the underlying semantics. Similar to current OS clipboards the Semantic Clipboard enables the source application to submit multiple representations of the same data, therefore increasing the probability that the data consuming application can process it. Using namespace or class URIs as identifiers, graphs defined in different vocabularies can be offered to target applications. The Semantic Clipboard can act as data mediator by looking up and executing mappings between these representations. As additional service the Semantic Clipboard completes graphs that contain unresolved references.

In a next step we will implement a prototype for the proposed architecture. With this prototype we will reconsider the issues presented in Section 5 by evaluating different use cases. Of particular interest will be the question if graphs should better be handled on the ontology or class level. The results of this evaluation should then prepare the ground for the integration of Semantic Web technologies into OS clipboards.

References

1. D. Brickley and L. Miller. FOAF Vocabulary Specification. `http://xmlns.com/foaf/0.1/`, July 2005.
2. J. Clark. XSL Transformation (XSLT) Version 1.0. *W3C Recommendation*, November 1999. `http://www.w3.org/TR/xslt`.
3. N. Freed and N. Borenstein. RFC2045: Multipurpose Internet Mail Extensions (mime) part one: Format of internet message bodies. IETF RFC, November 1996. `http://tools.ietf.org/html/rfc2045`.
4. H. Halpin, B. Suda, and N. Walsh. An Ontology for vCards. `http://www.w3.org/2006/vcard/ns\#references`, November 2006.
5. B. Nowack. The Web Clipboard. `http://www.sparqlets.org/clipboard/home`. Last visited: February 2007.
6. R. Ozzie. Live Clipboard - Wiring the Web. `http://www.liveclipboard.org`. Last visited: February 2007.
7. E. Prud'hommeaux and A. Seaborne. SPARQL query language for RDF. W3C Working Draft, October 2006. `http://www.w3.org/TR/rdf-sparql-query/`.
8. G. Reif, M. Morger, and H. Gall. Semantic clipboard - semantically enriched data exchange between desktop applications. In *Semantic Desktop and Social Semantic Collaboration Workshop at the 5th International Semantic Web Conference ISWC06*, Athens, Geogria, USA, November 2006.
9. World Wide Web Consortium (W3C) Semantic Web activity homepage. `http://w3c.org/sw`.