Personal Knowledge Mapping with Semantic Web Technologies

Matthas Hert, Gerald Reif, and Harald Gall

Software Evolution and Architecture Lab, University of Zurich
Binzmuehlestrasse 14, CH-8050 Zurich, Switzerland
{hert,reif,gall}@ifi.uzh.ch

Abstract: Semantic Web technologies promise great benefits for Personal Knowledge Management (PKM) and Knowledge Management (KM) in general when data needs to be exchanged or integrated. However, the Semantic Web also introduces new issues rooted in its distributed nature as multiple ontologies exist to encode data in the Personal Information Management (PIM) domain. This poses problems for applications processing this data as they would need to support all current and future PIM ontologies. In this paper, we introduce an approach that decouples applications from the data representation by providing a mapping service which translates Semantic Web data between different vocabularies. Our approach consists of the RDF Data Transformation Language (RDTL) to define mappings between different but related ontologies and the prototype implementation RDFTransformer to apply mappings. This allows the definition of mappings that are more complex than simple one-to-one matches.

1 Introduction

Today, the World Wide Web consists of several billion documents that are publicly accessible and serve as a rich source of knowledge. This has already made the Web a valuable resource for knowledge workers, but it has its limitations in exchanging data between software system if the meaning of the data has to be preserved. The explicit encoding of the semantics would ease the processing of that data and therefore enable new applications as well as increase the value of existing data. The Semantic Web [BLHL01] provides technologies to address these problems and we can observe a continuously growing popularity in the domain of Knowledge Management (KM) as described in [War06] and a special issue of IEEE Internet Computing [DLS07].

Personal Knowledge Management (PKM) also benefits from this development as ontologies exist to encode data from the Personal Information Management (PIM) domain. There are ontologies for contact data (e.g. FOAF,[1] vCard,[2] NCO,[3] SWRC [SBH+05]), event data (e.g. RDF Calendar,[4] SWRC), and wiki data (e.g. Semantic MediaWiki [KVV06]) that

[1] http://xmlns.com/foaf/spec/
[2] http://www.w3.org/2006/vcard/ns
[3] http://www.semanticdesktop.org/ontologies/nco
[4] http://www.w3.org/TR/rdfcal/

enable the representation of PIM data in RDF. However, Semantic Web technologies also introduce a new problem of heterogeneity as each party is free to use any existing ontology or define a new one to represent their application data. We can clearly observe this problem in the PIM domain where various ontologies exist that cover the same or strongly overlapping areas. It is unlikely that all these vocabularies will be replaced by one unifying ontology, but rather additional ones will emerge deteriorating the situation even more. This causes problems to applications that want to process Semantic Web data as they would not only have to support all currently available ontologies, but the applications would also need to be updated every time a new vocabulary emerges. Therefore, we see the need for a service that acts as a mediator between applications and Semantic Web data. This service decouples the applications from the concrete representation of the data by providing translations for data encoded in different but related ontologies.

To further motivate our approach, we present in this paragraph an example use case where RDF data should be exchanged between two PIM applications via a Semantic Clipboard [RLMG07]. Imagine that we want to add the birthdays of all persons in our address book to our calendar application. The address book encodes the contact data (including birthdays) in the vCard ontology, but the calendar application employs the RDF Calendar vocabulary and cannot process vCards. As a consequence, the Semantic Clipboard needs to transform the source data before it gets pasted to the target application. This mediation process is handled by our RDFTransformer component that runs locally as part of the Semantic Clipboard. The advantage of a local transformation service is that it does not depend on a central server and therefore ensures the privacy of the sensitive PIM data.

The contribution of this paper is an approach to bidirectionally transform RDF data between ontologies. The approach consists of three parts: (1) The mapping language called RDF Data Transformation Language (RDTL) to define correspondences between two ontologies; (2) the prototype RDFTransformer implemented as a library that enables the application of mappings; and (3) the stand-alone server application Remote Mapping Storage (RMS) to distribute existing mappings over the Web.

The remainder of this paper is structured as follows: Section 2 takes a brief look on related work in the area of ontology mapping. Section 3 compares the two basic approaches for mediating between applications and data that use different ontologies. Section 4 introduces our approach for transforming RDF data between different vocabularies, including our mapping language RDTL and the prototype implementations RDFTransformer and RMS. Section 5 summaries the results from our evaluation and Section 6 concludes this paper.

2 Related Work

The problem of mapping between ontologies can be split into two parts. First, the correspondences between matching elements have to be defined, either manually or automatically by alignment algorithms. Second, the mappings have to be applied on data to convert it from a source to a target format. These two parts so far received different amounts of attention from the research community.

A lot of effort has been put into the automated finding of corresponding concepts. Such approaches are manifold and they can be differentiated by the characteristics they use to detect a match. There are approaches that focus on linguistic and structural similarity (e.g. Cupid [MBR01]); some need the same set of instances encoded in both vocabularies and then analyze the resulting identical individuals (e.g. FCA-MERGE [SM01]); others investigate the mapping to a common reference ontology (e.g. IF-Map [KS03]). A combination of multiple techniques was also realized (e.g. OLA [EV03, EV04]). This enumeration is not exhaustive, there are other approaches as well as implementations to detect matching concepts. Their findings can also be used to define RDTL mappings.

In contrast, there was less work done in the representation of ontology mappings and their application in RDF data mapping tools. RDFTranslator[5] was developed as a tool for ontology development and lets the user define mapping rules that are used to translate RDF data from one vocabulary into another. Anchor-PROMPT [NM01] provides functionality for both finding and applying mappings on RDF data. It is implemented as a plugin for the ontology engineering tool Protégé[6] and therefore uses its native formats and GUI elements. MAFRA [MMSV02] is a framework for mapping distributed ontologies that covers the entire mapping process from automatically finding matches to the execution of mappings. Stecher et al. present in [SNN08] an approach for information integration on personal desktops. They use mappings to rewrite queries posed in a user-defined vocabulary to the ontologies of the information sources. Partial mappings are computed automatically and refined during query execution. However, the mappings are limited to simple one-to-one relationships and the queries to conjunctive combinations of triple patterns (i.e. triples where each of the subject, predicate, or object part can be a variable).

3 Query Rewriting versus Graph Transformation

Semantic Web applications typically use query languages to extract relevant parts from an RDF data set. If the ontology used to encode the data differs from the one employed by the applications, a mediation strategy is needed that translates between the two representations. There are two points in the mediation process where a translation approach can be applied. The first is the query that can be rewritten to match the target data. The second aims at the data by transforming the RDF graph to the vocabulary used in the application and therefore in the queries.

We opted for the RDF graph transformation in our approach due to four advantages it has over query rewriting. (1) Transformed data can be processed like any other Semantic Web data (e.g. reasoning before querying, applying rules), while the query rewriting approach is limited to querying the data. (2) The data transformation process needs to be applied just once per data set, whereas query rewriting must be performed for each query. In situations where one data set is queried oftentimes, the cumulated rewriting effort can exceed the one needed for data transformation. (3) Transforming data is a one step process after which

[5] http://wiki.corrib.org/index.php/RDFTranslator
[6] http://protege.stanford.edu

the data can be used natively, while query rewriting always needs two translation steps. First, the query has to be rewritten to the vocabulary of the target data and second, after its execution, the query results have to be translated back into the vocabulary of the source application. (4) The application of data transformations is simpler in situations where vocabularies are highly mixed, i.e., when a data set uses multiple ontologies. In the data transformation approach, the individual mappings defined to map from one source to one target ontology can be applied successively to transform the entire data set. This only increases the runtime but not the complexity of the approach compared to the case where the data is encoded in a single ontology. In the query rewriting approach it is unknown which parts of the data is encoded in what vocabulary. As a consequence, the original query has to be translated to every vocabulary occurring in the target data and each of these translated queries need to make heavy use of the OPTIONAL operator to ensure that the queries return the expected results. This not only increases the runtime but also the complexity of the approach with respect to the single vocabulary case.

At first, data accessible solely through a SPARQL endpoint seems to be a major limitation of the data transformation approach in contrast to query rewriting. However, SPARQL endpoints are also problematic to query rewriting as it is in general not known which vocabularies are used in the data exposed by the endpoint.

4 RDF Graph Transformation

In this section, we present our approach for transforming RDF graphs between ontologies. We first introduce in Section 4.1 our mapping language RDTL that is used to define correspondences between resources in a source and a target ontology. Section 4.2 gives an overview of our prototype implementations, the RDFTransformer and the mapping storage RMS, for bidirectionally translating RDF graphs.

4.1 Mapping Language RDTL

We analyzed various ontologies from the PIM domain to gather the requirements for our mapping language RDTL. We investigated how certain concepts are represented and how they can be mapped onto each other. Details about the analysis and the collected requirements can be found in [Her08]. In summary, the analysis resulted in the following requirements:

One-to-one Mapping: Most of the mappings will be simple one-to-one mappings, i.e., straightforward replacements of the property terms.

Typed Literals: Not all ontologies use datatypes for literal values, therefore they have to be added or removed during the mapping.

Nested Data: Ontologies are free to group related properties in a nested substructure or represent them individually. Support for extracting, creating, and converting of nestings is needed.

Literals/URIs: There are ontologies that represent certain properties as literals although their contents are actually URIs (e.g. email adresses). Creating real URIs from literals and vice versa is required.

Implicit Information: The same information can be represented differently so that it is stored explicitly in one ontology but only implicitly in another. A mapping should enable the extraction of implicit information.

Subject Types: Besides handling the translation of properties, every mapping also needs to adapt the type classes of the subjects.

Listing 1: Mapping document example

```
<?xml version="1.0" encoding="UTF-8"?>
<mappings xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="schemas/mapping.xsd">
  <namespace prefix="foaf">http://xmlns.com/foaf/0.1/</namespace>
  <!-- ... other namespace definitions ... -->
  <subject-group>
    <source-type>foaf:Person</source-type>
    <target-type>vCard:VCard</target-type>
    <simple-mapping>
      <source>foaf:birthday</source>
      <target datatype="xsd:dateTime">vCard:bday</target>
    </simple-mapping>
    <nested-mapping>
      <target>foaf:family_name</target>
      <target>foaf:givenname</target>
      <source-container name="vCard:n" type="vCard:Name">
        <source>vCard:family-name</source>
        <source>vCard:given-name</source>
      </source-container>
    </nested-mapping>
    <complex-mapping>
      <forward-mapping>
        <arg>prop:Start_date</arg>
        <arg>prop:End_date</arg>
        <source>fn:toDuration</source>
        <target>ex:duration</target>
      </forward-mapping>
      <backward-mapping>
        <arg>ex:start</arg>
        <arg>ex:duration</arg>
        <source>fn:toEndDate</source>
        <target>prop:End_date</target>
      </backward-mapping>
    </complex-mapping>
    <!-- ... other mappings ... -->
  </subject-group>
  <!-- ... other subject groups ... -->
</mappings>
```

We developed our ontology mapping language RDTL to meet these requirements. It uses a simple XML syntax to represent the mappings based on a XML Schema definition to enable validation of mapping documents and to ensure the correct order of matching source-target pairs. Listing 1 shows a small example of a mapping definition containing one example for each mapping type. The root element in a mapping file is called *mappings* and encloses the namespace definitions and the actual mappings grouped by the individual subject they apply to. RDTL supports prefixes to abbreviate long URI namespaces in the remainder of the mapping file (see line 4 for an example namespace definition). A *subject group* is defined for each class occurring in the source ontology and it encapsulates all mappings that are applicable to individuals (the subjects) belonging to that class. This enables the mapping of properties based on the context (the class of the subject) it is used in (e.g. the same name property is used for persons and organization in the source ontology but mapped to two distinct properties in the target vocabulary). Lines 6 to 36 represent a subject group definition containing individual mappings. First, on lines 7 and 8, the class type of the subject is mapped, where *source-type* contains the class name of the source ontology which is used to select the individuals during translation and *target-type* names the class associated to the individual after the mapping. Next, the mappings of the relevant properties are defined as source-target pairs, each belonging to one of the three supported mapping types: simple, nested, or complex mapping. A *simple mapping* is used for the one-to-one mappings that simply correlate a property from the source ontology with one from the target ontology. Simple mappings further provide features to add/remove a datatype, transform a literal value into another format, and convert between literals and URIs. Each feature is implemented as an XML attribute of the *target* or *source* element that specifies the datatype or the converter function, respectively. The example on lines 9 to 12 shows a simple mapping that adds a datatype to the mapped birthday property. The *nested mapping* is aimed at the requirement of nested data with support for extracting, creating, and converting nestings. Extracting implies that only the source ontology contains a nesting which is represented as a *source-container* element in RDTL. It contains the *source* elements that are matched to the *target* elements outside the container in their order of appearance. Lines 13 to 20 depict an example of a nested mapping that gets extracted. Likewise, creating implies that only the target ontology contains a mapping and the *target* elements are enclosed in a *target-container* element while the *source* elements reside outside the container. Converting a nesting implies that both container types are present and all *source* and *target* elements are encapsulated in their respective containers. Each container has a name attribute identifying the property that links a subject with an RDF container (e.g. an instance of rdf:Bag) and a type attribute that names the class of the container (the example uses the vCard ontology that links a subject via the vCard:n property to an rdf:Bag of the type vCard:Name containing the individual name parts). It is sufficient to define only one direction of a simple or nested mapping, the reverse mapping is created automatically enabling a bidirectional mapping with only half the definition effort. The *complex mapping* is the most flexible type of mapping as it allows to call external functions that can perform arbitrary operations on the underlying RDF graph, including the extraction of implicit information. This flexibility necessitates that both directions of the mapping have to be defined explicitly. Lines 21 to 34 contain a complex mapping consisting of a *forward-mapping* and a *backward-mapping* to reflect the two mapping di-

rections. Each mapping is made up of one *source* element that contains the name of the function to call, a *target* element with the name of a property that will link to the result of the function call, and optionally a number of *arg* elements that act as input parameters to the function. The comments on lines 35 and 37 indicate that the number of individual mappings as well as subject groups is unbounded. A detailed description of our mapping language with more examples can be found in [Her08].

4.2 Prototype Implementations

We implemented a prototype called RDFTransformer that can process and apply mappings defined in RDTL to transform RDF data encoded in one or more source vocabularies to a target ontology. It is realized as a library that can be embedded into other applications to equip them with mapping functionality. Figure 1 shows an overview of its architecture.

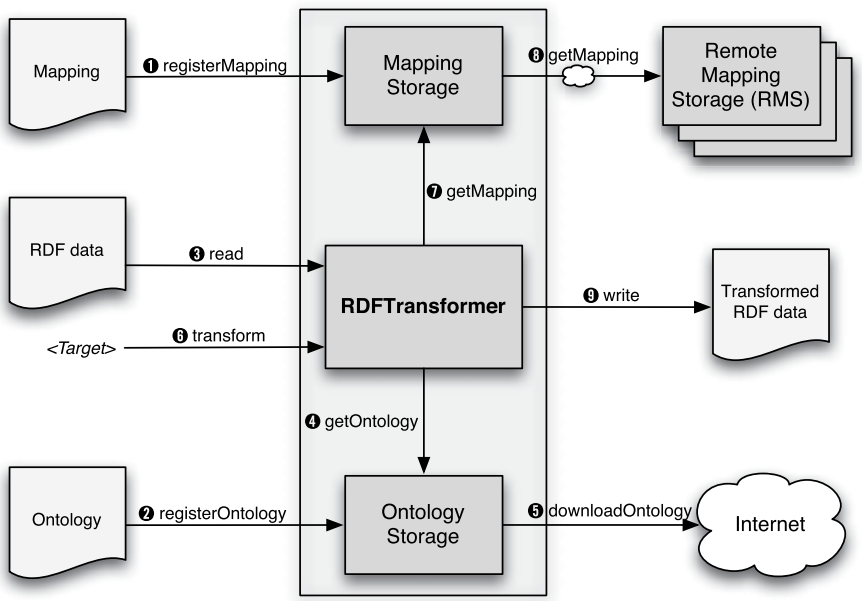

Figure 1: Overview and architecture of the RDF Transformer

The RDFTransformer is composed of three modules, the main module also called *RDF-Transformer*, the *Mapping Storage*, and the *Ontology Storage*. The Mapping Storage module is responsible for managing the mappings. Its main task are: providing a local interface to register new mappings (1); compiling mapping definitions to the internal format (a set of SPARQL CONSTRUCT queries); storing existing mappings; downloading missing ones from RMS servers on demand (8); and serving request from the main module (7). The

RDFTransformer uses a reasoner combined with ontology definitions to enhance the type information in the input data. This information is later used in the mapping process to select a subject group with appropriate mappings for each instance. Handling the ontology definitions is the task of the Ontology Storage module. Its responsibilities are: providing an interface to the developer for manually registering new ontologies (2); storing the definitions; automatically downloading missing ones from the Web (5); and serving requests from the main module (4). The RDFTransformer module provides the main interface for the transformation of RDF data into another vocabulary. It enables a three step transformation process: First, the input data is read either from a local file or a remote URL (3) and enhanced as explained before. Second, the URI of the target ontology is submitted (6) which triggers the RDFTransformer to analyze the input data and determine what mappings are needed. They are retrieved from the Mapping Storage (7) and applied to the input data. Third, a write command is issued to serialize the transformed data to a file (9). Both the input and the transformed data are cached so that the same data can be translated to multiple target ontologies and written to multiple files without the need to repeat the previous steps. More details about the architecture and implementation can be found in [Her08].

The Remote Mapping Storage is a self-contained server on the Web exposed via an XML-RPC interface. Its purpose is to centrally provide common mappings over the Web for other applications that use the RDFTransformer library. It stores compiled mappings and provides them on request to RDFTransformers. In addition, the RMS server offers functionality to check for new versions of mappings. The existence of independent RMS servers has the benefit, that RDFTransformer-enabled applications do not need to import all mappings manually but can receive them individually when they are demanded.

5 Evaluation

We implemented the RDFTransformer prototype described in the prior section as a proof of concept and for evaluation purposes. Realized as a library, it is meant to be embedded in other applications that need mapping functionality. On our website,[7] we provide a binary version of the RDFTransformer library for download that contains the library itself as well as all required third-party components, configuration and auxiliary files. The release also includes a simple command line interface to the RDFTransformer for testing purposes. It enables the user to easily interact with the prototype for registering new mappings defined in RDTL and applying existing ones to RDF data. For that purpose, this release is preconfigured to access the already defined mappings stored in our public RMS server.

We provide a public Remote Mapping Storage server as described in Sect. 4.2. It contains mappings from the PIM domain that we defined during our research on this topic. We currently offer the following bidirectional mappings: FOAF $\leftrightarrow$ vCard, FOAF $\leftrightarrow$ NCO, vCard $\leftrightarrow$ NCO, vCard $\leftrightarrow$ SWRC, SWRC $\leftrightarrow$ NCO, and SWRC $\leftrightarrow$ BibTeX. A detailed description on how to access and use this RMS server is available on our website.

[7] `http://seal.ifi.uzh.ch/RDFTransformer`

In order to make testing the RDFTransformer easier for interested parties, we also offer a simple Web application on our website that enables users to try it without any setup effort. It provides the same set of predefined mappings as our RMS server described in the previous paragraph, while the RDF input data can be supplied by the user. For detailed usage instructions see the description on our website. This Web application is only a user interface to collect the input data and display the results. The actual transformation is performed by our RDFTransformer library.

In a next step, we implemented a Semantic Clipboard that is able to transform the source data into another ontology before pasting it to the target application as motivated in Sect. 1. A second Semantic Clipboard with mapping capabilities, called KDE SemClip, was implemented in [Wol08] for the K Desktop Environment (KDE).[8] They extended the native system clipboard to support RDF data and modified existing applications to make use of the new transformation features. Their implementation is based on mappings defined in our mapping language RDTL and it also uses the Remote Mapping Storage as a source for predefined mappings.

Our experiences with these multiple implementations and applications of our approach showed the general usefulness of such a service as well as the applicability of our bidirectional mapping language RDTL.

6 Conclusion

We have presented an approach for mapping RDF data bidirectionally between different ontologies from related domains. The distributed nature of the Semantic Web and the growing adoption of its technologies in the domain of Personal Knowledge Management also exposes this area to the heterogeneity problem. Multiple ontologies are used to encode data from the same or overlapping domains making the development and operation of Semantic Web-enabled applications harder. Our approach contributes to the resolution of this problem by introducing the mapping language RDTL and the RDFTransformer library to decouple applications from the data representation. RDTL supports more diverse mappings than simple equivalency statements (*owl:equivalentClass* and *owl:equivalentProperty*) provided by OWL, while remaining easy to use. The RDFTransformer library enables the simple embedding of mapping functionality in PKM tools and it also showed the feasibility of our mapping language RDTL. The implementation of two Semantic Clipboards demonstrated the usefulness and applicability of our approach in handling RDF data encoded in different ontologies across application boundaries.

References

[BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, 2001.

[8] `http://www.kde.org`

[DLS07] John Davies, Miltiadis Lytras, and Amit P. Sheth. Semantic-Web-based Knowledge Management. *IEEE Internet Computing*, 2007.

[EV03] Jérôme Euzenat and Petko Valtchev. An Integrative Proximity Measure for Ontology Alignment. In *Proceedings of the Semantic Integration Workshop at the 2nd International Semantic Web Conference*, 2003.

[EV04] Jérôme Euzenat and Petko Valtchev. Similarity-based Ontology Alignment in OWL-Lite. In *Proceedings of the 15th European Conference on Artificial Intelligence*, 2004.

[Her08] Matthias Hert. RDF Graph Transformation - Bridging between Ontologies. Master's thesis, University of Zurich, 2008.

[KS03] Yannis Kalfoglou and Marco Schorlemmer. IF-Map: An Ontology-Mapping Method Based on Information-Flow Theory. *Journal on Data Semantics*, 2003.

[KVV06] Markus Krötzsch, Denny Vrandečić, and Max Völkel. Semantic MediaWiki. In *Proceedings of the 5th International Semantic Web Conference*, 2006.

[MBR01] Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm. Generic Schema Matching with Cupid. In *Proceedings of the 27th International Conference on Very Large Data Bases*, 2001.

[MMSV02] Alexander Maedche, Boris Motik, Nuno Silva, and Raphael Volz. MAFRA - A MApping FRAmework for Distributed Ontologies. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management*, 2002.

[NM01] Natalya F. Noy and Mark A. Musen. Anchor-PROMPT: Using Non-local Context for Semantic Matching. In *Proceedings of the Workshop on Ontologies and Information Sharing at the 17th International Joint Conferences on Artificial Intelligence*, 2001.

[RLMG07] Gerald Reif, Gian Marco Laube, Knud Möller, and Harald C. Gall. SemClip - Overcoming the Semantic Gap Between Desktop Applications. In *Proceedings of the 6th International Semantic Web Conference*, 2007.

[SBH+05] York Sure, Stephan Bloehdorn, Peter Haase, Jens Hartmann, and Daniel Oberle. The SWRC Ontology - Semantic Web for Research Communities. In *Proceedings of the 12th Portuguese Conference on Artificial Intelligence - Progress in Artificial Intelligence*, 2005.

[SM01] Gerd Stumme and Alexander Maedche. FCA-MERGE: Bottom-Up Merging of Ontologies. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, 2001.

[SNN08] Rodolfo Stecher, Claudia Niederée, and Wolfgang Nejdl. Wildcards for Lightweight Information Integration in Virtual Desktops. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, 2008.

[War06] Paul Warren. Knowledge Management and the Semantic Web: From Scenario to Technology. *IEEE Intelligent Systems*, 2006.

[Wol08] Tobias Wolf. KDE SemClip - Integration of the Semantic Clipboard into the K Desktop Environment. Master's thesis, University of Zurich, 2008.