Building Ontology-Based Annotations for Personal Information Management Systems – The SemanticLIFE Design and Implementation Case

Khabib Mustofa, A Min Tjoa
Institute for Software Technology and Interactive Systems
Vienna University of Technology
Favoritenstrasse 9-11/188, Vienna, Austria
{khabib, tjoa}@ifs.tuwien.ac.at

Abstract- **Since the introduction of the Semantic Web idea, many tools have been developed to harvest the advantage of machine-understandable data. Ontology and metadata are the two basic enabling requirements therefore. SemanticLIFE is a Personal Information Management system which deals with data and metadata originating from various sources. Being developed on the basis of Semantic Web technology standards, the system involves ontologies and represents metadata in RDF(S) or OWL.**

To enrich metadata for the system, semantic annotation is considered essential and hence this paper discusses the annotation approach implemented for SemanticLIFE. Due to the variety of data sources, annotation is not solely addressed to web pages, but also extended to emails, files and other supported data types. The annotation tool-module described in this paper provides users with an ontology-based way of producing relationship-metadata by either creating new instances or by specifying relationships amongst existing instances. This metadata creation process is facilitated by an ontology-supported navigation and the support of inference mechanisms on already existing metadata.

I. INTRODUCTION

The World Wide Web has made changes to the society and business by making information instantly and almost anywhere available. It enabled the possibility of transporting information from using physical to electronic means. No wonder that within just few years, it has become source of abundant information. During that time, with existing approach, it was considered sufficient to spread and exchange information using that media in which the recipient which can understand the information is *human*. This is unavoidable because the content were represented and encoded in natural language [3].

Later, it was expected and realized that in future machine can do jobs normally carried out by human and this requires the *machine* to understand what to do on what. Related to the web, the idea of Semantic Web was introduced [2]. On a machine understandable web, it is possible that the machine can determine what "documents are about". This can be accomplished by annotating entities mentioned in the documents, for example: persons, places, organizations and other entities. For the Semantic Web vision aiming at this goal, at least two classes of metadata: *ontology support* and *annotation* should be investigated extensively. Ontology support will maintain metadata of entities based on some domain ontologies and annotation is used to make reference to the entities' metadata.

Semantic web proposes to represent annotation in forms that bear semantic meaning, by which it is possible to identify concepts and relationships between concepts in a document. This implies that pure textual annotation about document-content is not sufficient for the purpose of machine understanding. Annotation is required to be based on ontology, so it has some understandable format and can bear semantic meaning. This ontology-based semantic annotation can enhance information retrieval and improve interoperability. From the information retrieval aspect, it is possible to perform the search process based on annotations because annotations can actually be regarded as part of the metadata,. It also allows resolving anomaly in searches, for example in the case when we have two or more words of similar spelling representing different concepts. Obviously for providing the interoperability, annotation which is based on common ontology can be substantially used to bridge the different information sources with their possible terminological mismatch.

Realizing the potential of the semantic web, many annotation tools were consequently developed to realize the above described goals. Some of them are independent tools (developed for general annotation) and some are plug-ins or embedded modules of bigger systems. Most of them are restricted to web pages but a few are trying to extend the scope to other forms of documents. A broad comparison of those tools is given in [4].

This paper discusses annotation system used in SemanticLIFE [1]. SemanticLIFE is a Personal Information Management which adopts the semantic web technology. Its main idea is based on the effective and efficient management of metadata related to documents or information collected from the various sources of someone's life (email, person of contact, web page, file, process state and notes/task/event (e.g. from Outlook)). In the

This work as part of the SemanticLIFE project is generously supported by ASEA-UNINET (ASEAN-EU Academic University Network)

implementation, it is based on ontologies and the use of semantic web technology standards, such as RDF(S) and OWL. Some details on the SemanticLIFE and its annotation is elaborated on section IV. Before that, section III discusses scenario examples and requirements. After the discussion of related work in section V, the paper is concluded by sketching the horizon of future work in section VI.

II. CONTRIBUTION

Different from most semantic annotation systems which are built for web pages as the primary focus of Semantic Web, SemanticLIFE deals with a variety of data sources,. Thus semantic annotation applies in SemanticLIFE for a whole bunch of supported data defined with their specific ontologies: emails, files, web pages and other classes. Annotation is also defined as a class in this framework of ontology, therefore it subsequently supports the building of *"annotations on annotations"*. An annotation's content can either take the form of textual annotation (for example when the user formulates a question or some comment) or in some form which already captures more semantic meaning, for example by adopting the basic construct of *rdfs:Statement*.

Using this approach the user can construct a statement by choosing subject, predicate and object and the annotation system will consult the ontology if it follows the rules and constraints defined.

III. SCENARIO AND REQUIREMENTS

The significance of the semantic annotation facility in the SemanticLIFE system can be seen as one important pillar forming the ideas of a system which support a life-long Personal Information Management. In this paper we will not discuss a complete scenario of the potential of SemanticLIFE, but we will restrict our description of a scenario due to issues which are relevant for building annotations based on ontologies. Consider that the system is used to store documents and information related to research or project work in which a user may be engaged. In this limited domain, the user will need to store various information and documents, such as: persons with whom he may collaborate, files (documents of proposals, publication papers, pictures video), emails, chat session, web pages browsed to find important materials for the project and calendar information (task, notes or events). For this cluster of information sources and documents, we need to create the potential meaningful and rich metadata. It would be tedious to feed the metadata completely manually, but based on today's state of the art it is illusionary that this could be done automatically.

To bridge such need, the system implements *both* manual and non-manual metadata collections. Manual metadata collection is carried out through annotation system, whereas the non manual way is performed through the data feeding modules: email feeds, web browsing session, contacts, calendar feed (tasks, notes, events), and file up-loaders. Each feeding module sends its data automatically and metadata is collected in a self-controlled manner. For example, data and metadata sent by the web-browsing-session module include: web page content, URL of the page, browsing (access) time, host or IP address from which browsing is carried out, referring page, last modification of the page and title of the page. The email-feeding module sends data and metadata of email content, sender, receiver (cc, bcc), sending date, receiving date, subjects and attachments information. Storing solely metadata collected from such feeding modules must be considered as not sufficient for a system which is expected to support semantic information retrieval. Many additional types of metadata need to be added to enrich the system. A web page can be annotated with content metadata such as: person, location, organization, event, project, product or other information. A similar approach is applicable also to emails, images and documents. In this scenario, the annotating process is actually a process of specifying associations between annotated objects, in this case: web pages, emails, images or documents with annotating objects. If the data about annotating objects are not yet available, the process would involve populating data of the annotating objects first. In this particular case, usually manual annotation is involved. Although semantic annotation seems more powerful than traditional textual description, textual annotation seems still playing its own role, to some extent. This kind of annotation is aimed more at human than machine control.

In systems adopting the semantic web concept, objects are classified by their assignment to some classes in the ontology framework. Relationships among objects or instances can be detected either by inference through the ontology-scheme or through instantiation of instances' properties. Up to this point, the requirements for the manual semantic annotation within the SemanticLIFE system seem quite straight forward. The manual semantic annotation must support:

1. **Object creation**. Because not all metadata can be collected from feeding modules, the module responsible for making annotations should support the creation of new objects. Creating new objects involves the following steps:
   - object identification. Each object created must be assigned to some unique identifier for later unambiguous reference.
   - object-class relationship definition. Each object must belong to a certain class in the ontology. This object is then called *instance* of the class.
   - Object-property relationship creation (property - value assignment). Objects are distinguished by characteristics expressed as property values, thus each object should be described by giving

values to the properties as defined in the ontologies

2. **Object-object relationship creation**. Relationship between objects can be defined after the objects are created. An RDF statement, in form of triple subject-predicate-object, is used to represent this relationship. This relationship plays an eminent role in semantic annotation
3. **Ontology support**. Ontology is used for navigation and inference in order to make the annotation process less complicated. The user interface for object creation is supported by navigation based on the underlying ontology structures. Some parameters in the interface are populated with deduced values by utilizing inferences on the properties' constraints and retrieving existing instances
4. **Textual annotation**. Although the content of annotation is free text, the representation of annotation follows the framework of semantic web in which reference to objects being annotated is defined through the URI.

IV. DESIGN AND IMPLEMENTATIONS

A. Design and Components Implementation

In SemanticLIFE, annotation constitutes either a module or subsystem which interacts with users. It consists of a graphical user interface supporting the above requirements. The user interface is not only designed to fulfil the basic task of manual annotation, moreover it provides a template-like construction where the user can input attribute values supported by the required ontology foundation, providing the assignment of allowable values and constraint checking.

The following components and essential related parts of the system are described in a nutshell:

1. **Class Lister**. This part provides the user with a list of classes defined in the ontology which are not populated by existing feeding modules. The metadata delivered are only the list of relevant classes. This facility is provided to support Object-Class definition of instance creation. To create a new instance, user first chooses a class type from the list (see Fig 1)
2. **Instance template**. Depending on the class selected when the user wants to create objects, a user interface as a template for the corresponding class will be generated. As a manual annotation tool, the interface distinguishes two control categories for attribute values: *text-box* in which the user can input some string value and *list-box/combo-box* where the user can select given values for the corresponding attributes. Such a list-box is populated by invoking queries and inferences on the existing ontology model. Using this approach, the user will be substantially supported in creating annotations and the data integrity can be enhanced because it will considerably reduce the tendency of making redundant annotations or duplicate object creations.
3. **Objects Relations Creator**. By means of this interface, the user can indicate a statement on the content of annotation instance by specifying subjects, predicates and objects. The mechanism of denoting such statement is simple, the user just drag the selected object from the object browser panel and drop it to the annotation panel (see Fig 2). For example, if the user wants to make a statement that persons X, Y, Z have collaborated in event E, the user is required to drag item X, Y and Z to the subject part of the interface, select predicate (link property) `collaborateIn` from the options listed in a combo-box and drag object E of class `Event` to the object part. When these values are submitted, the ontology will be consulted to ensure whether the constraints are fulfilled. In this example the constraints are domain and range constraints, where the domain of property `collaboratedIn` is of class `Person` and its range is of class `Event` or `Publication` (see Fig 4). Such an ontology support and check mechanism is an important step to maintain data integrity, since the easy drag and drop process inhere the risk that the user will breach the constraints defined in the ontologies.
4. **Free Text Annotation**. Using a schema concept like the Object-Object relationship, it is possible to store meaningful textual annotations. The reference to the object being annotated is performed via URI and the annotation text is treated as literal object. As an example, a user can annotate an email by making an RDF-like statement with the URI of the email as the value of property `subject`, property `hasComment` as the value of property `predicate` and the text "This email is very important" becomes the value of property `object`.
5. **Simple Object Crawler**. Although this part is much closer to the storage part, the crawler's task is very much related to annotation. The crawler does the following "simple" checking tasks:
   – checking the instances in the model whether they have unavailable necessary metadata or not. In the ontology we define some properties to be of higher priority in terms of their values instances. If a property is defined as necessary, then its instantiation is expected, although it is not an absolute requirement. When the crawler finds some unavailable metadata, it informs the user through annotation subsystem to assign some value to these properties.
   An example of such a property is the name of a person. Anyway, the determination of the grade of the properties' importance depends

very much on the ontology creator. A property `belongTo` of class `EmailAddress` can be considered important when class `Person` does not have property `mbox` which relates the class `EmailAddress` with class `Person`.

the existence of instance of Y (of type class K). If the same instance of Y already exists, then it is not necessary for the crawler to trigger the creation of that instance. In case that the instance does not yet exists it will trigger an action for creating that instance. One important

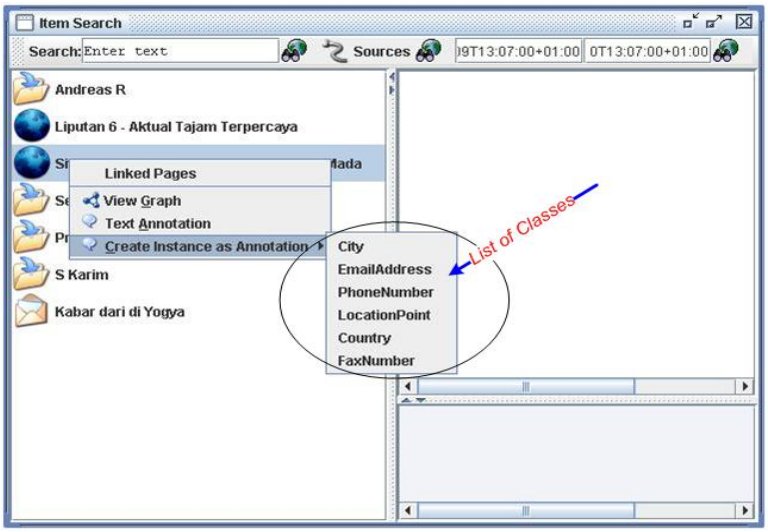

Fig. 1. Interface for choosing textual annotation or create new instance

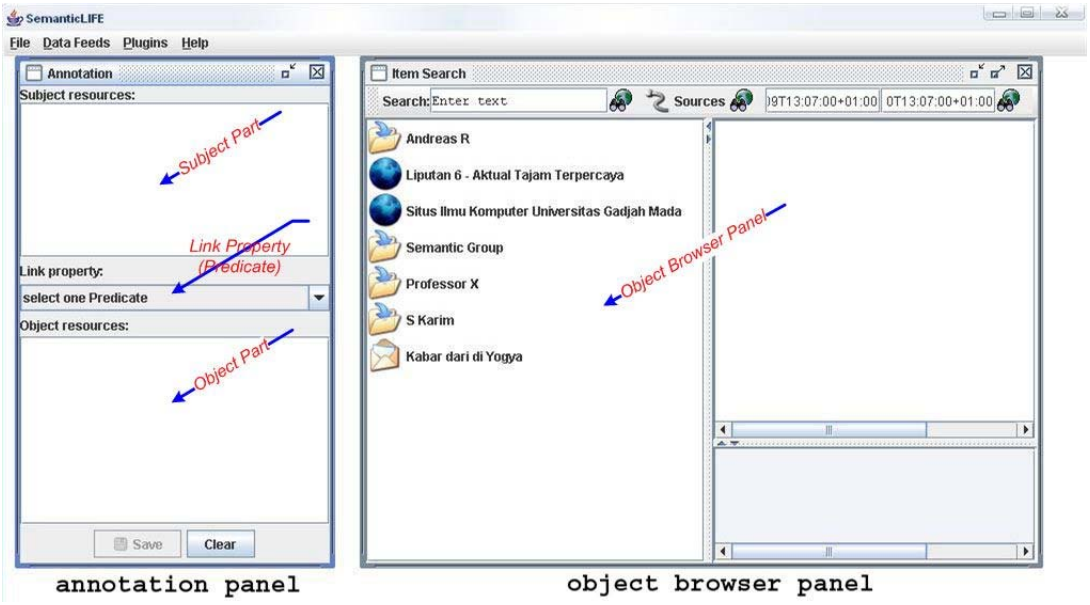

Fig. 2. Interface for making objects - objects relationship

– checking the existence of instances of some classes which are possibly contained in instances of other classes. As mentioned previously, some classes are populated by specific feeding modules. Feeding module X may feed metadata of some instance of class C in which the metadata can be used to instantiate an object of class K. When an instance of C is created, the crawler will check

showcase example is of an email-feed. When an email is fed, it contains metadata about sender and receivers which are email addresses. If, in the ontology, class `EmailAddress` has property `mbox`, `personalName` and `belongTo`, then some of the metadata from email can be used to populate an instance of `EmailAddress`, especially for attribute `mbox` and

`personalName`. Subsequently, when another email involves the same `mbox` as sender or receiver, such instance of `EmailAddress` is not necessarily created again.

B. Ontology

In the ontology, annotation is defined as a class concept. An annotation can contain comments or relationships among instances. For this purpose, class `Annotation` is defined in the ontology as shown in Fig. 3, also a class `Statement` which consists of the three properties: `subject`, `predicate` and `object`.

A statement can be constrained or non-constrained (general) ones. For a general statement, user can make any assertion by choosing subject, predicate and/or object. Constrained statement requires some of the properties to take restricted values according to some predefined criteria in the ontology. This is achieved by declaring it as subclasses of Statement plus some restrictions on the properties. Consequently, this is implemented as a named-class. Fig 4 shows a definition for a named-class `Collaboration` as a subclass of `Statement` with constraints for its subject, predicate and object. The followings show a brief example for the case.

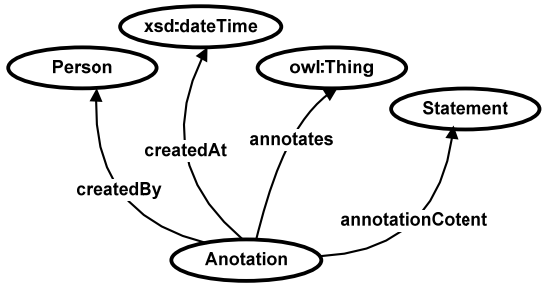

Fig. 3. RDF-graph of Class `Annotation`

- A statement whose predicate is "`hasComment`" has constraint that the object is of type string (XML string).
- A statement to denote collaboration will have the constraints that the subject is of type `Person`, the predicate has value `collaborateIn` and the object of type `Event` or `Publication`.

Properties used for relating objects in such a statement are declared as sub-properties of property `relations`. In Fig 2, the items in the *Link Property* combo-box take these values.

With the basic construct of a sentence, describing or annotating resources becomes more expressive, for example:

- an email discussing an appointment of person X and Y to have a meeting on 31-01-2006 in University Hall can be annotated with two sentences: "X, Y meet at 31-01-2006" and "X,Y meet in the University Hall". Here, X and Y act as the subjects, "meetIn" and "meetAt" are defined properties in the ontology.
- an image can be annotated with a sentence "Alex kiss Annie" in which "Alex" and "Annie" are instances of person in the ontology.

Another ontology issue is the choice of properties for labelling the GUI, as can be observed on the object browser panel of Fig. 1 and Fig. 2. In RDF, a label for the short description of a resource can use `rdfs:label`, but not all classes implement this property and in some cases users prefer to use the value of some property as the label for them, instead. Copying each value of such property to assign a value of `rdfs:label` means duplicating things which is not recommended. As an alternative, we define such a property to be subclass of `rdfs:label`.

```
<owl:Class rdf:ID="Collaboration">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Statement"/>
    </rdfs:subClassOf>
    <rdfs:label rdf:datatype=
"http://www.w3.org/2001/XMLSchema#string">
        Collaboration</rdfs:label>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty
             rdf:about="#subject"/>
        </owl:onProperty>
        <owl:allValuesFrom>
          <owl:Class rdf:about="#Person"/>
        </owl:allValuesFrom>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:allValuesFrom>
          <owl:Class>
            <owl:unionOf
             rdf:parseType="Collection">
            <owl:Class rdf:ID="Event"/>
            <owl:Class rdf:about="#Publication"/>
            </owl:unionOf>
          </owl:Class>
        </owl:allValuesFrom>
        <owl:onProperty>
        <owl:ObjectProperty rdf:about="#object"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:hasValue>
         <owl:ObjectProperty
            rdf:ID="collaborateIn"/>
        </owl:hasValue>
        <owl:onProperty>
          <owl:FunctionalProperty
              rdf:about="#predicate"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
```

Fig. 4. Definition of class `Collaboration`, a subclass of `Statement`

In Fig 2, each object is shown with its proper labelling. Objects represented with globe icon are of type `Webpage` and envelope is of type `Email`. Analogous to the idea of property `relations`,

properties intended to be used as label are declared to be sub-property of `rdfs:label`. For the case of webpage and email in this example, property `title` of webpage and `subject` of email are appropriate, as in the following snippet:

```
<owl:DatatypeProperty rdf:ID="title">
    <rdfs:subPropertyOf
rdf:resource="http://www.w3.org/2000/01/rdf-
schema#label"/>
    <rdfs:domain rdf:resource="#Webpage"/>
</owl:DatatypeProperty>
```

Using this approach, with simple SPARQL query, we can get labels for various type of object easily:

```
SELECT distinct ?item ?type ?title WHERE {
{?item rdf:type ?type}
{?item ?titlep ?title}
{?titlep rdfs:subPropertyOf rdfs:label}
}
```

V. RELATED WORKS

There are numerous semantic annotation tools available [4]. Among these tools, CREAM [5], Medcertain [6], [7], Annotea [9] and the annotation for image collections [8] inspired the development of SemanticLIFE's annotation subsystem. CREAM offered an annotation for web pages with some very interesting features in order to make it user friendly. CREAM is especially designed for annotating and authoring web pages. Medcertain included the idea of using RDF statements for the modelling the content of annotation. Annotation for image collections [8] uses Semantic Web standards for annotating art media (images) which enables user to describe what is depicted in the media based on terms from some libraries.

In SemanticLIFE, the annotation subsystem combines some feasible approaches used by some existing tools and extends the annotation for a whole variety of information sources. Furthermore, it shows also that semantic web standards such as RDF and OWL can be broadly used for managing general metadata, which are not necessarily restricted on the web environment but also extendible for desktop application in the case of Personal Information Management Systems.

VI. CONCLUSION AND FUTURE WORKS

In this paper we shows the manual semantic annotation approach used in the SemanticLIFE system. Manual annotation plays an eminent role in all systems where the human intervention as an end-user for commenting various data sources is required.

The SemanticLIFE system, at this stage, is primarily focused on personal use, and is implemented in a bundle as a desktop-centred application. Up to this stage, web pages, emails, files and documents have been shown feasible to be annotated in a generic way. From the users' point of view, semantic annotation can be utilized using statements in form of triples which provides an easy and expressive way of feeding metadata.

Up to this stage, we present a manual annotation on documents or information source as a whole. Even-though some systems already support annotation on a specific part of document, such as Annotea [9] for web pages, we consider such an implementation as a next step for a further extension SemanticLIFE. The fact that various documents are supported needs more time to implement it.

REFERENCES

[1] M. Ahmed, H. H. Hanh, S. Karim, S. Khusro, M. Lanzenberger, K. Latif, M. Elka, K. Mustofa, N. H. Tinh, A. Rauber, A. Schatten, N. M. Tho, and A. M. Tjoa, "SemanticLIFE -a framework for managing information of a human lifetime," in *Proceedings of the 6th International Conference on Information Integration and Web-based Applications and Services*, September 2004.

[2] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities," *Scientific American*, May 2001

[3] S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, K. S. McCurley, S. Rajagopalan, A. Tomkins, J. A. Tomlin, and J. Y. Zien, "A case for automated large scale semantic annotation," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 01, no. 01, pp. 115–132, 2003.

[4] V. Uren, P. Cimiano, J. Iria, S. Handschuh, M. Vargas-Vera, E. Motta, and F. Ciravegna, "Semantic Annotation for Knowledge Management: Requirements and a Survey of the State of the Art", in *Journal of Web Semantics*, Vol 4, Issue 1, 2006

[5] S. Handschuh and S. Staab, "Authoring and annotation of web pages in CREAM", in *The 11th International World Wide Web Conference*, May 2002

[6] P. Cross, "Proposal for rdf annotation classes," December 2000, retrieved in November 2005. [Online]. Available: http://www.medcertain.org/ilrt/annotation prop.html

[7] P. Cross, L. Miller, and S. Palmer, "Using RDF to Annotate the (Semantic) Web," in *Knowledge Markup and Semantic Annotation Workshop Notes, 1st International Conference on Knowledge Capture*, pp. 21–33, 2001.

[8] L. Hollink, G. Schreiber, J. Wielemaker, and B. Wielinga, "Semantic Annotation of Image Collections," in *Workshop on Knowledge Markup and Semantic Annotation*, 2003.

[9] J. Kahan and M.-R. Koivunen, "Annotea: An open RDF infrastructure for shared Web annotations," in *Proceedings of the 10th International World Wide Web Conference*, pp. 623–632, 2001.