Ontology-Based Querying of Linked XML Documents

Lule Ahmedi* Georg Lausen
Institut für Informatik, Universität Freiburg
Georges-Köhler Allee, Geb. 51
79110 Freiburg, Germany
{ahmedi,lausen}@informatik.uni-freiburg.de

ABSTRACT

We investigate the Lightweight Directory Access Protocol (LDAP) that provides a powerful means to link distributed data collections into an entity searchable as a single collection. Besides the linking mechanism, LDAP offers a rich collection of modeling primitives required to express ontologies as a common searchable interface. Encouraged in addition by the fact that XML has become the de facto standard for information interchange on the Internet, LG ACCESS, our global querying system, combines ontologies and XML into a unified LDAP-based framework to improve the power for accessing related XML data in the network. We describe a global query evaluation strategy based on expressive links among entities that are spread across different local or remote servers. Also, the way ontologies are annotated to hold such network-aware links along with semantic annotations is shown. Built on grounds of the standard and well-established LDAP technology instead of the technologies that are still in the process of adoption by the community, our system is an ideal candidate for applications that need to interact with semistructured databases at a global extent.

1. INTRODUCTION

In recent years the proliferation of XML-based systems has dramatically increased to the point of requiring the redesign of existing software and even the creation of new paradigms to handle semistructured data. At the same time, since the conception of the LDAP protocol version 3 in 1997 [35], the use of lightweight directories to store a variety of information has been steadily gaining momentum. Today, many universities and research centers use LDAP servers as a means to manage information about their members, organizations, networks, etc., and companies like Netscape or Microsoft offer LDAP support even in their Internet browsers. Among a large number of directory servers [21], an implementation that supports running LDAP against relational databases is now available [15].

Recently, Tim Berners-Lee's vision of a "Semantic Web" [5] in which a new form of Web content, that is meaningful to computers and linked up to be easily processable by machines on a global scale, is attracting a great attention in the community. From the point of view of databases, the Semantic Web can be seen as a globally linked multidatabase, where links pointing from one database object to the other can readily carry out sophisticated tasks for users/applications when querying. To bring a meaning to the content of data, ontologies can be used to hold a kind of semantic annotations along with the outgoing links that serve to relate data. As regards the aspect of handling queries for the case of the aforementioned circumstances (interlinked documents), the Semantic Web technology presents yet an evolving infrastructure.

On the other side, the similarities between LDAP and the XML "native" model, i.e. DOM [16], makes the former an ideal candidate for querying XML-based sources without the need to incur in cumbersome transformations. Also, the development of XML technology, as well as the adoption of the still evolving standards, like XLink [12] or XPointer [11], is not as well established as LDAP.

Therefore, instead of using the technologies that are still in their infancies, we have opted to base our querying system on LDAP technology, and expect to leverage its strengths for the purpose of querying semistructured data networks at a global extent. Decisive thereby is its tight connection to network and distribution channels, that, by design, offers distribution capabilities far superior from those found in traditional multidatabase systems [35]. Links that relate remote data can make particular use of such capabilities, enabling efficient query processing of those data regardless of the breadth of their distribution.

Our main objective of combining ontologies and XML into a unified LDAP-based framework for global querying is to improve the power for accessing XML data, relieving the information searcher from considering the distributed allocation of data and their semantical and structural heterogeneity in distinct sources. This is not the case when using any of the query languages so far proposed (XQuery [7], XML-QL [13], etc.) for accessing XML documents. We illustrate the distinctions by an example, both in XQuery and in our approach, and outline the improvements we offer.

Example 1 *Assume that we are interested in integrating data about provinces from two geographic databases, the* TERRA *and GlobalStatistics databases which are available in XML form from [28]. Fragments of each of these XML*

*The work of this author is supported by the Deutsche Forschungsgemeinschaft, Aktenzeichen La 598/4-1.

Semantic Web Workshop 2002 Hawaii, USA

sources and their DTDs are shown in Fig.1.

```
<gs>
<province name="Nordrhein Westfalen">
 <population>17816079</population>
 <area>34077</area></province>
<province name="Bayern">
 <population>11921944</population>
 <area>70546</area></province>
  :
</gs>
```

```
<terra>
<province name="Tirol"
 abbrev="TIR" pop="586700"/>
<province name="England"
 abbrev="ENG" pop="47100000"/>
<province name="Bayern"
 abbrev="BAY" pop="10973700"/>
 :
</terra>
```

Figure 1: Excerpt of XML data

In XQuery syntax, the query would be the following:

```
FOR $prov_gs
       IN document("gs-europe.xml")//province,
    $prov_terra
       IN document("terra-europe.xml")//province
WHERE $prov_terra/@name = $prov_gs/@name AND
      $prov_terra/@pop > 10000000
RETURN <province abbrev=$prov_terra/@abbrev>
         <name>{$prov_terra/@name}</name>,
         <population>{$prov_terra/@pop}
                              </population>,
         <area>{$prov_gs/area}</area>
       </province>
```

and returns the province *elements as above, with their* name, population, abbrev *taken from* Terra, *and* area *taken from GS.*

In our system, we achieve the same result if we submit a query to the ontology where we have to specify only the entities we want to have at the output, and not where they have to come from and how they should be combined, as in the following (using XPath):

```
//StateGeopolitical[population > 10000000]
```

given that provinces are named as StateGeopolitical *in the ontology.*

In this paper, we present the global query evaluation strategy behind this idea targeted to the ontologies model, putting the emphasis on the network features of the approach that works particularly well in the case of a highly distributed data topology. The rest of the paper is organized as follows: Section 2 gives a brief overview of the capabilities offered by LDAP servers. Section 3 provides an explanation of the overall architecture of our system, whereas Section 4 goes in short through the modeling issues used, leaving the discussion about links as the key constructs of this model for Section 5. A global query evaluation strategy based on links is elaborated in Section 6, and then continued with the emphasis on its efficiency in Section 7. Finally, Section 8 compares our approach with other systems, and concludes the paper.

2. LDAP OVERVIEW

An LDAP [35, 20] server can be viewed as a semistructured database system, specialized for the operations performed on the Internet, that is, simple and fast remote read operations that occur much more frequently than writes (that are in general local), and support for the classical client/server architecture.

The LDAP data model consists of two components: The *directory schema* defines a finite set of classes, their organization in the schema hierarchy, together with their content, attributes and datatypes. The *directory instance* contains a finite set of entries organized in a forest, where each entry (1) has a non-empty set of (possibly) multi-valued attribute-value pairs $e(a, v)$ that conform to the schema definition; and (2) belongs to at least one class which is given by the (mandatory) attribute objectClass.

The naming model defines how to arrange entries into an instance hierarchy based on their names and how to refer to any particular entry. Giving a unique name to any entry in the instance hierarchy allows to refer to any entry unambiguously. The unique name of an LDAP entry is its *distinguished name* (dn). It is formed by enumerating all the individual names of the parent entries back to the root of the hierarchy. Reading the entry's dn, e.g., (cn=capital,cn=Country,o=Ontology), one can trace from the entry capital itself through the Country back to the root o=Ontology of the tree. In this work we are mainly interested on how to refer to individual entries in the hierarchy and how to handle referred entries when searching.

The functional model determines the operations that can be performed on the directory server. The search operation acts in a similar way a database query does. It is invoked by a search request defined as a combination of the following four components: The *base* denotes the distinguished name of the entry in the directory instance where the search will start. The *scope* can be base, if the search is to be restricted to just the first node, onelevel, if only the first level of nodes is to be searched, or subtree, if all nodes under the base should be considered by the filter expression. The *filter expression* is a boolean combination of atomic filters of the form $(a\ op\ v)$, where a is an attribute name, op is a comparison operator, and v is an attribute value. The *projection* defines the set of attributes to be returned by the query.

Referral Mechanisms. There are two kinds of referral mechanisms in LDAP we make use of in our framework, i.e., aliases and smart referrals.

An *alias* is an entry that contains the dn of another entry in the same server. If the search result contains an alias, that entry is *replaced* by the entry the alias points to through its dn, i.e., by its aliased entry.

Example 2 *Consider the following alias entry that describes capitals at the meta-data level of the geographic data introduced in Example 1*

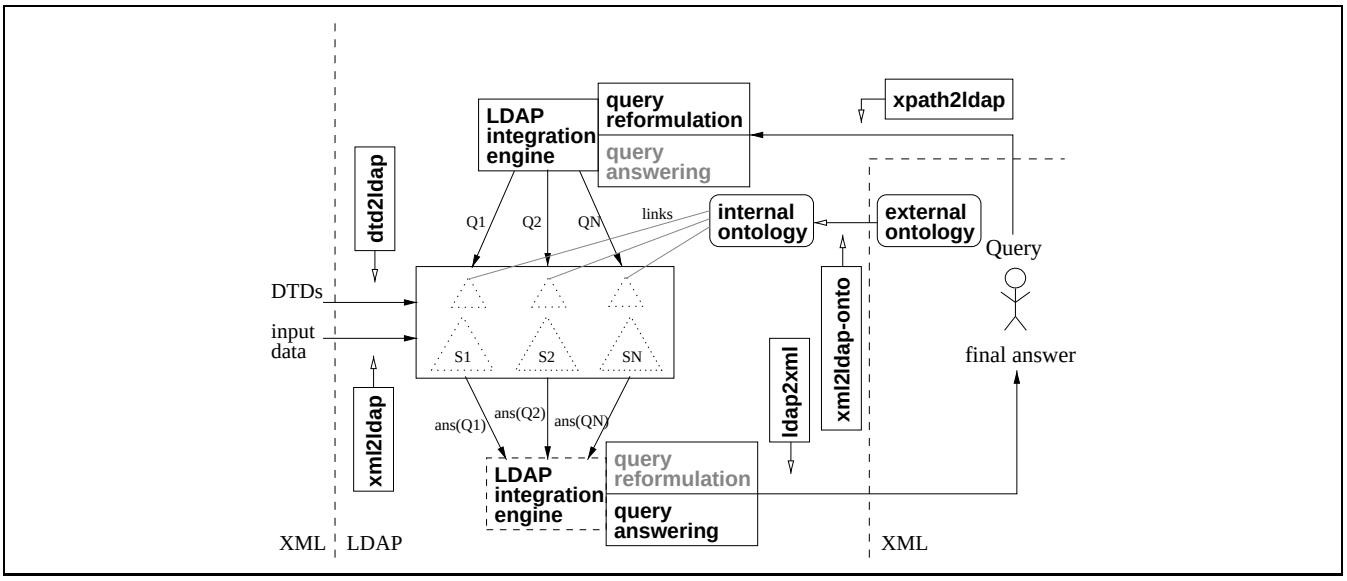

Figure 2: The system architecture

dn: cn=has_capital,cn=Country,o=Body,
ou=Ontology,o=LGAccess,dc=top
name: has_capital
objectClass: alias
aliasedObjectName: r=capital,c=country,c=terra,
ou=srcSchema,o=LGAccess,dc=top

It states among others that if the entry appears in the answer set, it is to be replaced by its aliased entry, i.e., the entry whose dn is r=capital,c=country,c=terra,...,dc=top.
For example, the following query that looks for a concept describing capitals in a given ontology:

Q = (cn=has_capital,cn=Country,o=Body,ou=Ontology,
o=LGAccess,dc=top ? subtree ?)

would instead of the has_capital alias defined above result in its aliased entry

dn: r=capital,c=country,c=terra,ou=srcSchema,
o=LGAccess,dc=top
name: capital

that is a capital concept as described at the source schema level.

A *smart referral* is, in a way, a generalization of an alias. It is an entry in the directory instance that refers to one or more entries in the same *or in another* LDAP server. References (ref attributes) are in the form of LDAP uniform resource locators (URLs). An LDAP URL [18] is of the form ldap://host/searchExpr, where searchExpr is the search request of the form base?projection?scope?filter to be sent to the contacted server. Smart referrals are handled in the same way as the aliases. E.g., the following reference points to the entries that satisfy a filter (indep_date > 1990) in a tree rooted at cn=Country,o=Ontology and located at the ServerA.LGAccess.org host:

ldap://ServerA.LGAccess.org/cn=Country,o=Ontology??
sub?(indep_date > 1990).

In XML terms, aliases are local, similar to the XML ID/IDREF mechanism, whereas smart referrals are global and resemble to the XLink/XPointer linking mechanism. The use of aliases and smart referrals for the global querying purposes is investigated in Section 5.

3. THE OVERALL ARCHITECTURE

In order to be capable of processing XML data, any common global querying framework based on the ontologies model is required to support the definition and management of schema information (e.g., the Document Type Definition (DTD) that describes the structure of an XML document), and the actual integrated contents (i.e., the XML documents).
Our framework provides, via an ontology definition, the required features to seamlessly process both types of data, as detailed in Section 6. Figure 2 shows the architecture of LGAccess, our LDAP-based global querying system. LGAccess stands for Lightweight Global Access system, a non-intrusive extension of the traditional LDAP server modified to meet the needs of global querying in the Web. It proceeds as described in the following.

An (*external*) *ontology* of the domain of interest available in the Web is incorporated in our system such that it is taken in its original representation, namely XML, and transformed into the corresponding LDAP-modeled ontology by the *xml2ldap-onto* component. Then, for each available XML source and its DTD, a corresponding representation in LDAP is generated by the *xml2ldap* and *dtd2ldap* components, respectively (for details see [27], [1]). Further, as part of the application design, the correspondences between the entries in the ontology and those in the source schemata are established by means of the LDAP referral mechanisms. Global queries are evaluated by the *LDAP Integration Engine*, composed of the *Query Reformulation Module* and the *Query Answering Module*. The user formulates a query (in XPath) over the *external ontology* and poses it to the system. The *xpath2ldap* component transforms the query from XPath to LDAP. The *Query Reformulation Module* rewrites it into the terms of the source level queries by using the knowledge derived from the referrals and the semantical definitions found in the *internal*

```
<ontology><header> ... </header>
  <body-definition> ...
    <role-def><role name="name"/><range>STRING</range>
              <cardinality>1</cardinality></>
    <role-def><role name="has_capital"/>
              <subrole-of><role name="has_city"/></subrole-of>
              <range><concept name="City"/></range>
              <inverse>is_capital_of</inverse></>
    <concept-def><concept name="GeopoliticalEntity"/>
      <role-constraint><role name="name"/><key>YES</key></role-constraint>
      <role-constraint><role name="population"/></role-constraint>
      <role-constraint><role name="area"/></role-constraint></>
    <concept-def><concept name="Country"/>
      <subconcept-of><concept name="GeopoliticalEntity"/></subconcept-of>
      <role-constraint><role name="has_capital"/></role-constraint></>
    <concept-def><concept name="StateGeopolitical"/>
      <subconcept-of><concept name="GeopoliticalEntity"/></subconcept-of>
      <role-constraint><role name="abbrev"/></role-constraint></>
  </body-definition>
  <instances> <!-- here comes the derived XML view --> </instances>
</ontology>
```

Figure 3: Geography ontology in XML

ontology. These queries are evaluated separately for each source by the LDAP server. Then, the results are combined by the *Query Answering Module* taking into account the rewriting previously formulated by the *Query Reformulation Module*. Finally, *ldap2xml* transforms the result from LDAP into XML.

This paper focuses on the *LDAP Integration Engine*. These components use some modeling bases that are briefly described in the next section. Section 5 then concentrates on modeling constructs that particulary contribute to these component facilities, i.e. links.

4. MODELING BASICS

Ontologies. Ontologies play a crucial role in our framework representing a common conceptual model for all users of a domain (concepts, roles, concept taxonomy, etc.). Built upon LDAP, they offer support for global querying (data integration) characteristics we are interested about in this paper along with the support for the standard ontology design requirements. The modeling and technical grounds of LDAP, their tight connection to network and distribution channels offer capabilities not present in prevailing data integration approaches, as will be detailed in the next section. We partition the ontology component into an *internal* ontology to be represented in the LDAP middleware in LGAccess, and an *external* ontology which resides as an XML source at a given, accessible location on the Web. Also, the *xml2ldap-onto* component is introduced in the system to provide a mapping between the conceptual entities of the internal LDAP ontology and those marked up externally in XML.

A detailed description how to model ontologies, both in LDAP and in XML, as well as a mapping between them is given in [1]. An example XML geographic ontology is sketched out in Figure 3.

Data Modeling. In our system, an arbitrary XML document is represented in LDAP according to the definitions made in [27] for data representation. In this representation, *elements* and *attributes* are modeled by LDAP classes. The meaning of each of the attributes in these classes (Figure 10), the oc, oid, name, order, and value attributes, is also explained there.

Example 3 *For our running example, we assume that the system already contains:*

- *the Geography ontology whose StateGeopolitical concept definition with name, abbrev, population, and area roles is depicted within a solid line framed box in Figure 4, and*
- *the* Terra *and GlobalStatistics data sources and their DTDs whose Province element definitions are depicted within dashed framed boxes in the same figure.*

5. REFERRALS AS INTEGRATION MEANS

When the modeling of the existing domain ontology, the schemata of the sources that are subject of integration and their data is done, the substantial task of the modeling process, that of establishing the relationships between those models follows.

The LDAP referral mechanisms introduced in Section 2 offer a good basis for expressing links. Aliases are well-suited for specifying local links. But, in a real-world integration scenario, data sources and their schemata may be available locally or distributed across several directory services in the network. In addition, an entry in an ontology may relate to more than one entries in available data source schemata. Smart referrals are appropriate to define potentially multiple links in a distributed directory topology.

We annotate ontologies in the middleware (internal ontologies) of our system to hold such links. Network-enabled ontologies present the gist of the discussion concerning the

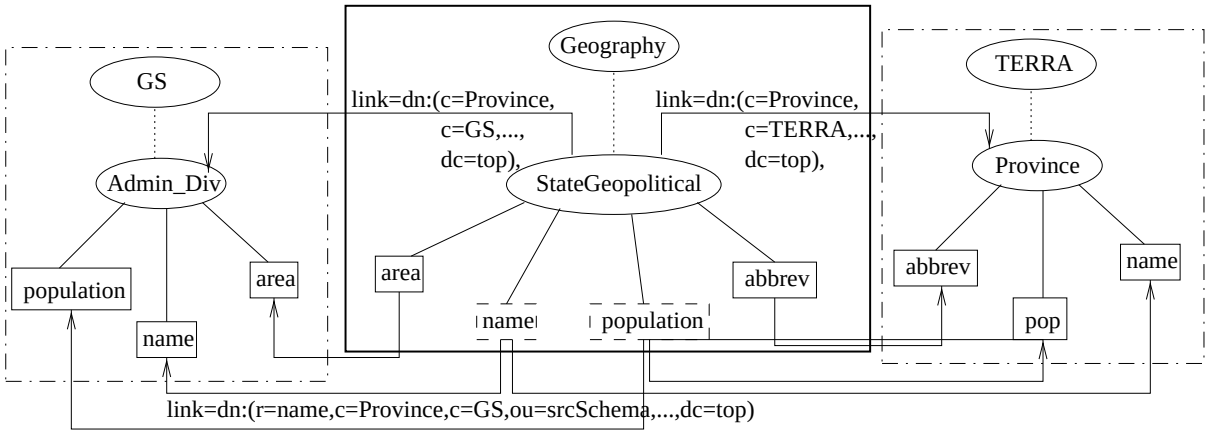

Figure 4: The GS and TERRA source schemata embedded in the Geography ontology

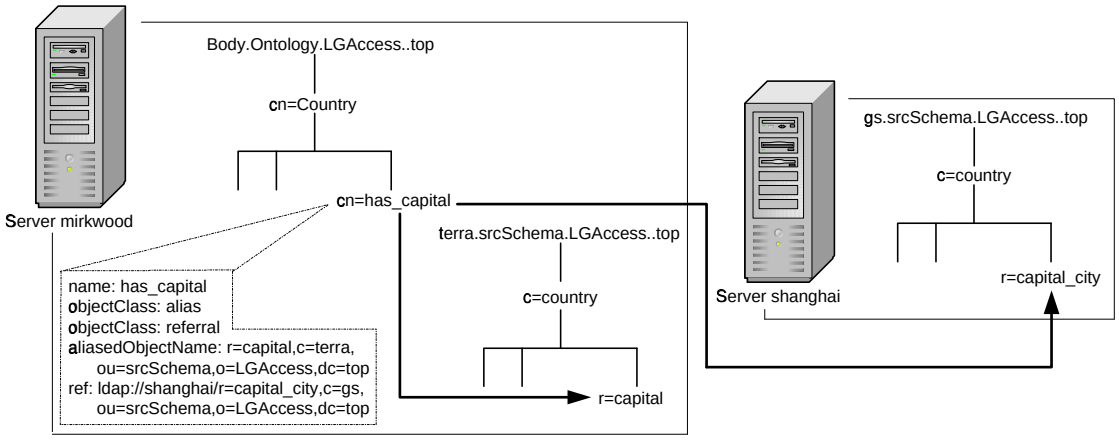

Figure 5: An example of specifying local and remote links in LDAP

modeling framework. Thanks to them, separate partitions of the whole model (source schemata and their data) in the network can be easily hooked together into a single conceptual view through the use of links.

As regards the interpretation of links when a query is issued to the system, LDAP's approach of handling aliases and smart referrals is quite basic. During the query evaluation in our system, the assigned links allow to relate any global queries performed over the ontology with their corresponding source level queries in a relatively easy fashion.

Figure 4 provides a graphical representation of the way our example ontology and the corresponding source schemata are related by means of the link attributes. The exact specification of these attributes in the LDAP instance of another fragment of our geographic example (Example 3) to be used later in the paper is given in Figure 5.

6. QUERY PROCESSING

XPath [9] is a W3C recommendation for addressing node sets of *an XML document*. Most XML querying languages are built on top of XPath as a core language. The LGAccess system supports querying in XPath by deploying a slightly modified version of the *xpath2ldap* translator developed as a part of the HLCaches project [27].

Furthermore, the meta-information of LDAP ontologies regarding a global semantical and structural definition over several distinct XML documents - their schemata - can be used by the system for evaluating XPath queries to extract relevant data from *the related XML documents*.

6.1 XPath into LDAP Query Translation

There are two groups that can make use of ontology definitions in the LGAccess system. One are information searchers who do not substantially care about the semantic descriptions of entities as described by the ontology and would rather be satisfied with a kind of a *view* containing only structural elements of the domain and keeping the internal layout of the view across several different sources and/or ontologies transparent to them. The other group includes users/applications who are interested in the meta-data definitions in the *ontology*. It would be nice to query related XML documents and the global meta-information about them in one format, but this looks difficult unless a standard XML ontology model and a proper language for accessing it is conceived by the Semantic Web community. See [22, 33, 31, 34] for the ongoing work on consolidating a proper query language for meta-data.

XPath as a path-oriented language isn't the proper way to query an ontology description. The XML ontology tree cannot directly be used to derive an XPath query against an XML instance. Instead, we propose to use an *XML view* of the ontology which represents a higher level abstraction

of the domain of interest released from accurate semantic descriptions [1]. Hence, the ontology model does not explicitly include instance data from XML documents being integrated. It provides a pattern for picking up those data, i.e. XML view, which is enclosed within `<instances>` tags in the ontology specification (c.f. Figure 3). In this way, for *accessing the data* in LGAccess, the information searcher formulates a query by using the view as a pattern. For an example (reduced) XML view, see Figure 6.

```
<Geography><GeopoliticalEntity>
   <Country> ...
      <car_code/><has_capital/><indep_date/></>
   <StateGeopolitical>
      <name/><population/><area/><abbrev/></>
</GeopoliticalEntity></Geography>
```

Figure 6: Geography view in XML

On the other side, for *accessing the meta-data* in LGAccess using XPath, a straightforward way is to query the underlying model (LDAP), regardless of the surface syntax.

An XPath query that derives both, the underlying model of the related instance data and their meta-data in LDAP ontologies requires a slightly different algorithm when translated into LDAP from the one used in [27] that is at best illustrated through an example.

Example 4 *If we apply the algorithm to the following XPath query (see Example 1 for the abbreviated syntax)*

$$Q_{XPath} = /descendant :: StateGeopolitical \\ [child :: population > 10000000]$$

our xpath2ldap *convertor produces the translated LDAP query* $Q = (M, R)$ *where* M *is the main query,* R *is the refinement query [27], and:*

$$M = (dn(root)?subtree? \\ (name = "StateGeopolitical")),$$
$$R = (dn(StateGeopolitical)?onelevel? \\ (\&(name = "population")(value > 10000000))).$$

6.2 Query Reformulation

Armed with the modeling formalisms and with the XPath into LDAP query translator, we are now ready to tackle the problem of handling global queries. In general terms, a query performed on the ontology must be reformulated into source-level queries. See [25, 26, 32] for other data frameworks that also support query reformulation.

In our framework, the query reformulation task involves rewriting the original query formulated over the ontology into single-source queries by taking into account the ontological constructs:

- the link attributes in both concepts and roles,
- the semantic descriptions (subconceptOf, inverseOf, ...), and
- the inter-ontology relationships (synonymOf, hypernymOf, hyponymOf)

The link attributes carry the information that relates the data residing in multiple XML documents and are hence *the* constructs that are investigated for query reformulation in the sequel. For example, the query Q_{XPath} in Example 4 is formulated relative to the XML view shown in Figure 6. It is translated into LDAP yielding the query Q. In order to process such a query in our system, the Reformulate algorithm depicted in Figure 7 is used. The details of each step of the algorithm are given below.

Relevant Source Discovery. Since the cost of accessing an information source over the network is significant, the main optimization offered is the minimization of the number of external information sources that need to be accessed in order to answer the query.

The link property of pointing to the namespace contained on the same or a different server is a suitable means to delegate the query context expressed by *base* to the new context nodes $base_{sch_i}$ - one per each local/external relevant source. This category of links is referred to as baseLinks. Thus, $baseLink_i(base_i, base_{sch_i})$ is used to denote the existence of a connection between entities $baseLink_i$ from the ontology and $base_{sch_i}$ from a source level schema (DTD). Step 1 does the identification of the relevant sources from a given data repository. It involves examining the baselinks associated with the *base* entity appearing in Q and extracting the corresponding host information $baseLink_i.host_i$ for each of them $(i = 1, M)$.

For our example query and the geographic model depicted in Figure 4, the resulting set of baseLinks is as follows:

$$L_{base} = \{baseLink(StateGeopolitical, Province), \\ baseLink(StateGeopolitical, Admin\_Div)\},$$

giving $\{terra, gs\}$ as the set of relevant sources.

Query Decomposition. The query decomposition involves finding the set of source level search requests

$$Q_{src} = \{Q_i : Q_i = (base_i, scope_i, filter_i), i = 1, M\}$$

such that each of the components, *base*, *scope*, and *filter* in the original query Q is substituted by its corresponding associate, $base_i$, $scope_i$, and $filter_i$ in the source level schemata respectivelly. The process of substituting original *base* and *scope* is trivial thanks to the assigned semantics to the baseLink attributes being suitable for this kind of mapping and the easy of managing and processing those links in LDAP.

The major part of the algorithm deals with finding the filter component of its dedicated query Q_i, i.e., $filter_i$. Step 2 reduces the filter of the original query to $filter_{ont}$ by omitting all predicates dealing with non-ontological attributes, i.e., value checking predicates. Then it evaluates this query over the ontology, ignoring for a while the existence of the underlying sources and their schemata. The result is a set *absTargets* of so-called *absolute target entries*. The link attributes assigned on these entries, called targetLinks are parsed and their aliased and/or referred entries are extracted in Step 3 of the algorithm. We call such entries *potential targets* since they do still not represent the final targets we were looking for originally. They are entries found in the general context, i.e. without (yet) taking into consideration the context as restricted by the *base* specification in the original query; they belong to the source schemata, not to the source

Algorithm (Reformulate(Q, Ont, S_{sch}, S_{src}))

Let $Q = (base, scope, filter)$ be a global query posed over the ontology $Ont.$, and
S_{sch}, S_{src} be the set of source schemata, source data respectively.

/* STEP 1. Discovering relevant sources */
for $base$, find the set of links L_{base}, such that:
$L_{base} = \{baseLink_i(base, base_{sch_i}) : base \in \{Q \cap Ont\}, base_{sch_i} \in S_{sch_i}, i = 1, m\}$

/* STEP 2. Finding the set of absolute targets */
/* reduce the filter by discarding any value constraints, e.g. $(value > 1000000)$ */
$filter_{ont} = Reduce(filter)$
$Q_{ont} = (base, scope, filter_{ont})$
$absTargets = FindTargets(Q_{ont}, Ont) = \{absTarget_1, ..., absTarget_j, ..., absTarget_m\}$.

/* STEP 3. Finding the set of potential targets */
/* assume that for each absTarget, there exists at most one potTarget per rel. source */
for each $absTarget_j \in absTargets$, find the set of links L_{target}, such that:
$L_{target} = \{targetLink_i(absTarget_j, potTarget_{sch_{ij}}) : absTarget_j \in Ont,$
$potTarget_{sch_{ij}} \in S_{sch_i}, potTarget_{sch_{ij}}(a_{pt_j}, v_{pt_j})\}$.

/* STEP 4. Building the source level search requests */
$Q_{src} = \{\}$
for each $baseLink_i \in L_{base}$ {
/* group baseLinks & targetLinks by the host they belong to */
$potTarget_{sch_i} = \{\}$
for each $potTarget_{sch_{ij}} \in L_{target}$ {
if $(potTarget_{sch_{ij}}.host_{pt} == base_{sch_i}.host_b)$ {
$potTarget_{sch_i} = potTarget_{sch_i} \cup potTarget_{sch_{ij}}\}\}$
/* generate a new source level search request and add it to Q_{src} */
$Q_i = BuildSourceQuery(Q, baseLink_i, potTarget_{sch_i})$ }
$Q_{src} = Q_{src} \cup Q_i$

return $Q_{src} = \{Q_1, Q_2, ..., Q_n\}$

Figure 7: Reformulation algorithm

Algorithm (BuildSourceQuery(Q, $baseLink_i$, $potTarget_{sch_i}$))

Let $Q_i = (base_i, scope_i, filter_i)$ be the source level query to be found.
$base_i = base_{sch_i}$
$scope_i = scope$ // simply overwrite it
$filter_i = \{\}$
$potTarget_{sch_i} = \{potTarget_{sch_{ij}} : j = 1, n\}$
for each $(a_f\ op\ v_f) \in filter$ of Q {
/* replace v_f with the corresponding one from the potTargets */
j=1
do {
find $(a_{pt_j}, v_{pt_j}) \in potTarget_{sch_{ij}}$ where $a_{ptj} = a_f$
if $((a_f\ op\ v_{pt_j}) \notin filter_i)$ {
$filter_i = filter_i \cup (a_f\ op\ v_{pt})$ }
j++
} while $(potTarget_{sch_{ij}})$ }

return $(base_i, scope_i, filter_i)$

Figure 8: Reformulation algorithm (BuildSourceQuery procedure)

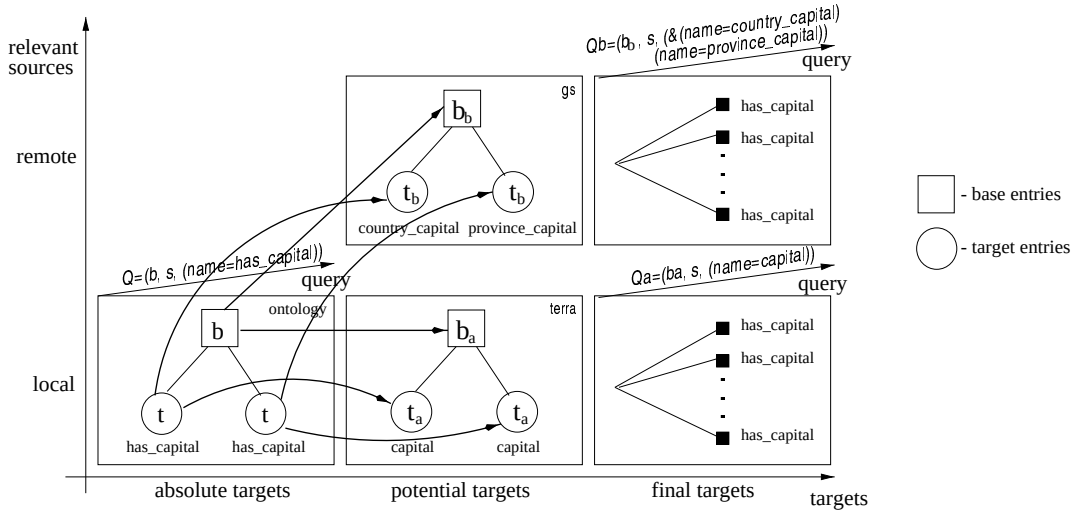

Figure 9: The proceeding of an reformulation scenario

data. Step 4 is the gist of the algorithm and the one that builds the source level search requests. After the baseLinks and the targetLinks are grouped by the baseLinks according to the host they belong to, they are sent as the arguments to the *BuildSourceQuery* function (c.f. Figure 8) which in turn generates a reformulated query definition in terms of its entries for each baseLink. Interesting is the way this function reformulates the filter component: For each filter predicate $(a_f\ op\ v_f)$ in Q, it generates a corresponding predicate $(a_f\ op\ v_{pt})$. If the set $potTarget_{sch_i}$ of potential targets contains more than one $potTarget_{sch_{ij}}$ entries, then for each $potTarget_{sch_{ij}} \in potTarget_{sch_i}$ a corresponding predicate $(a_f\ op\ v_{pt_{ij}})$ is generated if it differs from the ones generated from the previous $potTarget_{sch_{i1}}, \ldots, potTarget_{sch_{i(j-1)}}$ potential targets. At the end, the resulting predicates for a given $potTarget_{sch_i}$ are related with the boolean "&" operator.

In our case, applying Steps 2, 3, and 4 of the algorithm for the TERRA and the GS sources which are identified as relevant in Step 1 results in decomposing the original (global) query Q into the following source level queries respectively:

$$\begin{aligned} Q_{Terra} &= (dn(terra.root)?subtree?(name = Province)),\\ &\quad (dn(Province)?onelevel?\\ &\quad (\&(name = pop)(value > 10000000))\\ Q_{GS} &= (dn(gs.root)?subtree?(name = Admin\_Div)),\\ &\quad (dn(Province)?onelevel?\\ &\quad (\&(name = population)(value > 10000000))) \end{aligned}$$

Remind (Section 5) that links refer to the entries contained in the source schemata, not to the data source entries. Therefore, the task of redirecting the generated source-level queries from the source schemata to the respective data sources is also part of the reformulation process.

Example 5 *Consider the query*

$$\begin{aligned} Q_{dist} = (cn = Country, o = Body, ou = Ontology,\\ o = LGAccess, dc = top\\ ?\ subtree\ ?\ (name = has\_capital)) \end{aligned}$$

given that the has_capital fragment of our model is as in Figure 5.

The (de-)composition trace of this query wrt. different target trees and relevant sources is illustrated in Figure 9, giving an insight into the reformulation algorithm in general at the same time.

6.3 Query Answering

A set of nodes obtained from evaluating an XPath expression in our system is an unordered collection of nodes *with duplicates*. Remind that the XPath Recommendation specification defined by the W3C expects an unoredered collection of nodes *without* duplicates at the result set. As regards the ordering, the LGACCESS system goes beyond the original XPath specification, and can be asked to return a sorted collection of nodes at the output based on some attribute value [19]. Duplicate nodes arise when more than one sources contain the requested data. To eliminate them, the result nodes are exposed to the following modifications:

Renaming: The attributes that also appear in the query use `targetLinks` to map back from the source level namespace into the ontology vocabulary. In the case of Example 5, result nodes stemming from the TERRA source replace their actual name attribute values, country_capital and province_capital, with has_capital, those stemming from the GS source replace capital with has_capital as well.

Detection/Rejection: Duplicates, i.e., entries with the same value(s) for the value attribute are identified. Among them, those that come from the source believed to be more reliable (up-to-date) are chosen. The rest is discarded, avoiding redundant result data.

A *flat* answer set (list) of nodes does not fit the idea of hierarchically organized collections of XML data in the World Wide Web. Due to its hierarchical namespace, LDAP is ideally suitable for managing the tree-granular answer sets. Given the aforementioned arguments, we address next the problem of obtaining trees of depth 1 at the output of the system that mirror to a certain degree the concept-role relationships of ontologies. That means, if the user asks for a concept, its children (roles) are also attached to the result, as well as the other way round. This format of returning

results contributes considerably to enriching the answer set with multisource information, preserving at the same time the intended simplicity (lightweight) of our approach, ensured among others by the easy of "reading" results from the end-users. These pretensions become perceivable in the example below.

Again, for the case of the StateGeopolitical example, according to Figure 4, the set of links L found in Terra and GS sources determines that the answer set of Q_{Terra} and that of Q_{GS} are to be merged on the name node (see the line in Figure 3 that defines name as key). As a result of such a merge operation, the StateGeopolitical one-depth trees are composed as depicted in Figure 10. Comparing the individual Province answer set of Q_{Terra} together with that of Q_{GS} with the combined StateGeopolitical answer set, it is obvious that the latter is richer for the attributes abbrev and/or area, for at least some of the trees in the result set.

6.4 Discourse

Given the current state of the system, we assumed that each link, be it baseLink or targetLink, can be specified in the ldap://host/dn form, allowing the entries to be referenced by their distinguished names. Using links in the extended ldap://host/dn?projection?scope?filter form would allow for a much richer way of referencing entries and remains as future work. These can then be used as a means for resolving the structural discrepancies across sources.

Important to note is the time complexity of our reformulation algorithm. The overhead of the algorithm is mainly dictated by the task of reformulating the filter. Due to the assumption that for each absTarget entry, say M of them, there exists at most one corresponding potTarget per relevant source, say N of them, it is obvious that the algorithm is linear in $M \times N$. Note that composed filters in a query do not cause an increase in complexity because they are solely a matter of no other but the LDAP query engine itself.

Besides decomposition, further reformulation procedures not considered here are generalization, specialization, inversion, etc. based on the ontological primitives like subconceptOf, inverseOf, synonymOf, etc. See [1] for a detailed description of specifying these contructs in LGAccess.

7. EXPERIMENTAL RESULTS

A preliminary version of the system is already available and providing very promising results.

Consider again the Example 5. We used an excerpt of the freely available CYC Geography ontology on the Web at [24] that contains among others information about geopolitical entities, i.e., countries, states, cities, etc. For the instance data, we used the Terra and the GlobalStatistics data sources. We located the Geography ontology as well as the Terra data source at the local mirkwood server, and the GlobalStatistics data source at the remote shanghai server. We first submitted one query per source separately as if we knew the naming contexts of each of the sources and their relationship to our final global intension. Then, we posed a global query against interlinked sources such that a result set equal to the union of the previous result sets is obtained by following the local and remote links as depicted in Figure 5. Surprisingly, the time needed to evaluate the global query nearly equals the sum of the times needed to evaluate each of the single source queries separately, one after the other in series without taking advantages of possible parallelism (see Figure 11). Notice that, when performing the former, the user's expertise on a given domain and his familiarity with the available data repository is no longer required.

Another interesting measure is the time spent when searching within the network in cases where not all servers contain information relevant to answer a given query. A gain in efficiency is achieved when querying the entire network without broadcasting the query to every repository, i.e., without spending time to contact irrelevant servers.

These results, combined with the fact that the system is able to implicitly glue nodes belonging to different answer trees without incurring in any additional work from the user side, makes our system an ideal candidate for the usage in practice for global querying scenario of XML data in the Web.

Host Server(s)	Result Entries	Evaluation Time
mirkwood (local)	183	8.926s
shanghai (Intranet)	510	10.984s
mirkwood & shanghai	693	20.879s

Figure 11: The run-time requirement analyses

8. RELATED WORK AND CONCLUSION

For those familiar with XML, the question of using XLink/XPointer for specifying the links and adapt an existing XML query language to handle them in a proper way for any given distributed topology of XML documents comes natural. Actually, there is not yet an official definition about the interpretation of those links when queries occur, nor a system implementation that supports link specifications for querying in any way. [29] proposes a model where XML links are thought to serve to delegate a subtree at the point the link is defined in a document, but the approach aims not on freeing the user from the usually tremendous query definition task in case of a multisource environment (see Section 1). These facts, as well as the efficiency of the LDAP-based system to store and process XML data [27] and internet-wide pointing links (Section 7) argues for the deployment of LDAP instead of the emerging XML technologies in a middleware in our system.

Although extensive work has been done in the area of modeling ontologies for data integration, most researchers use logic-based languages like description logics [6], or F-Logic [23] to describe them and benefit from the reasoning abilities of logic systems. Our approach, on the other hand, uses a relatively simple model with a clear syntax that allows us to provide a unified formal framework to be used for both, ontology definition and information integration using the standard and well-established LDAP technology in the middleware.

Furthermore, due to the hierarchical and flexible structure of LDAP, we can very easily integrate not only structured data, but also semistructured data, as opposed to systems like Sims [2] or Observer [30] that are limited to the expressive power of their ontology description languages and are only able to deal with relational and flat file databases. Not even systems like Florid [17], Ontobroker [10], or Momis [4], which combine the reasoning capabilities of logic systems with the expressive power of an object-oriented model are able to describe XML data, as naturally as we do in our system. The lack of a tree-like modeling structure forces them to map XML into an artificial structure not particularly well-suited for such graphs, whereas our model is based

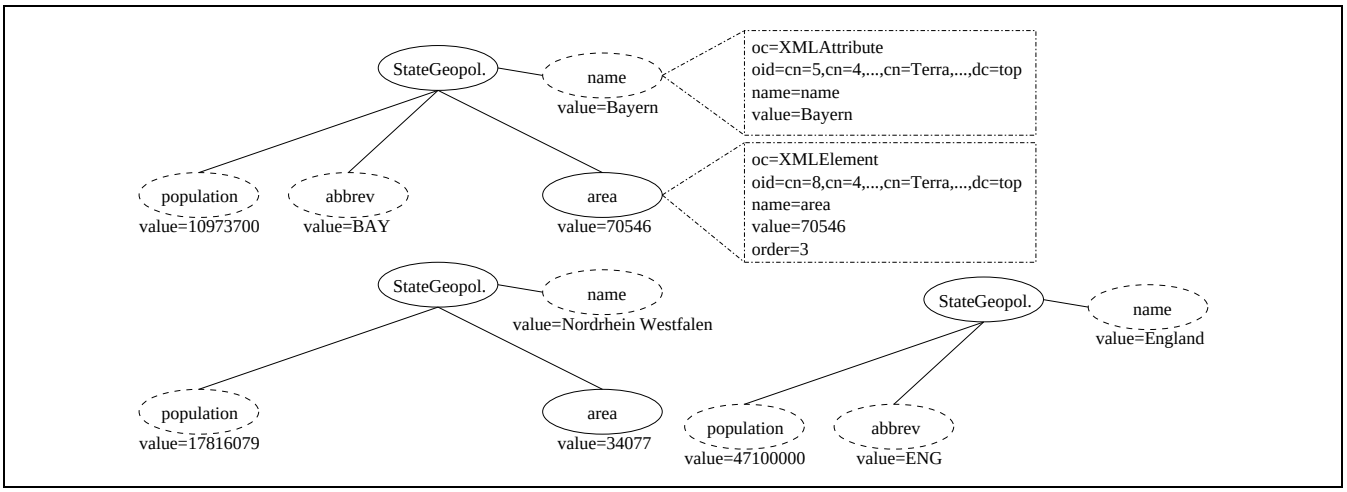

Figure 10: The result data

on a tree structure that resembles that of XML. TSIMMIS [14] addresses the above deficiencies by taking a similar approach to ours, namely using the OEM model to describe mediated views and sources, as opposed to our LDAP-based data model, which offers, by design, additional benefits derived by the nature of the LDAP directory services.

Some other approaches, like [3, 8] use XML as their common model for data exchange and integration, but donot consider the use of ontologies as an integral part of the system. Our LDAP-based model, on the other hand, provides seamless integration with ontologies and DTDs, which allows us to scale our system both, at the source level and at the user level.

Conclusion. In this paper, we have presented an approach for combining XML and ontologies into a unified LDAP-based framework that provides a common querying interface to XML data by means of powerful network-aware links that annotate ontologies and point to the instance data to be retrieved. A query evaluation strategy that supports this idea is explained in detail. Given the ubiquity of XML on the Web, our system will be useful for applications that need to interact with semistructured databases in a global sense. It can be applied to accommodate the namespace changes and mergers that are inevitable as organizations evolve, and to allow application designers to set up "search paths" for collecting results from multiple servers. It provides obvious advantages with respect to more classical integration approaches because of the simplicity, coherence, and uniformity of the LDAP model.

9. REFERENCES

[1] L. Ahmedi. Directory-Based Ontologies for Integrating XML Data. Intl. Workshop on Foundations of Models and Languages for Data and Objects (FMLDO'01), Viterbo, Italy, Sept. 2001.

[2] Y. Arens and C. Knoblock. SIMS: retrieving and integrating information from multiple sources. *SIGMOD Record*, 22(2):562–563, June 1993.

[3] C. Baru, A. Gupta, B. Ludäscher, R. Marciano, Y. Papakonstantinou, P. Velikhov, and V. Chu. XML-based information mediation with MIX. *SIGMOD Record*, 28(2):597–599, 1999.

[4] S. Bergamaschi, S. Castano, and M. Vincini. Semantic integration of semistructured and structured data sources. *SIGMOD Record*, 28(1):54–59, 1999.

[5] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, May 2001.

[6] A. Borgida. Description logics in data management. *IEEE Transactions on Knowledge and Data Engineering*, 7(5):671–682, 1995.

[7] D. Chamberlin, D. Florescu, J. Robie, J. Siméon, and M. Stefanescu (Eds). XQuery 1.0: An XML Query Language. W3C Working Draft, June 2001. `http://www.w3.org/XML/Query/`.

[8] V. Christophides, S. Cluet, and J. Simeon. On wrapping query languages and efficient XML integration. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 29(2):141–152, 2000.

[9] J. Clark and S. DeRose. XML path language (XPath) version 1.0. `http://www.w3c.org/tr/xpath`, 1999.

[10] S. Decker, M. Erdmann, D. Fensel, and R. Studer. Ontobroker: Ontology based access to distributed and semi-structured information. In DS-8: Semantic Issues in Multimedia Systems. Kluwer Acad. Publisher, 1999.

[11] S. DeRose, E. Maler, and R. D. J. (Eds). XML Pointer Language (XPointer) Version 1.0. `http://www.w3.org/TR/xptr/`, Sept. 2001.

[12] S. DeRose, E. Maler, and D. Orchard. XML Linking Language (XLink) Version 1.0. `http://www.w3.org/TR/xlink/`, June 2001.

[13] A. Deutsch, M. F. Fernandez, D. Florescu, A. Y. Levy, and D. Suciu. XML-QL: A Query Language for XML. In *WWW The Query Language Workshop (QL)*, Cambridge, MA, Dec. 1998. `http://www.w3.org/TR/1998/NOTE-xml-ql-19980819/`.

[14] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. D. Ullman, V. Vassalos, and J. Widom. The TSIMMIS approach to mediation: Data models and languages. *Journal of Intelligent Information Systems*, 8(2):117–132, 1997.

[15] O. Group. Openldap server. LDAP implementation available from `http://www.openldap.org/`.

[16] W. D. W. Group. Document object model

specification. http://www.w3.org/DOM/, Dec. 1997.

[17] R. Himmeröder, P. Kandzia, B. Ludäscher, W. May, and G. Lausen. Search, analysis, and integration of Web documents: A case study with FLORID. In *Proc. Intl. Workshop on Deductive Databases and Logic Programming (DDLP)*, pages 47–57, UK, 1998.

[18] T. Howes and M. Smith. RFC 2255: The LDAP URL format. Available from ftp://ftp.isi.edu/in-notes/rfc2255.txt, Dec. 1997. Status: Proposed Standard.

[19] T. Howes, M. Wahl, and A. Anantha. RFC 2891: LDAP control extension for server side sorting of search results.

[20] T. A. Howes, M. C. Smith, and G. S. Good. *Understanding and Deploying LDAP Directory Services*. Macmillan Technical Publishing, 1999.

[21] Innosoft. LDAP world implementation survey. Available from http://www.innosoft.com/ldap_survey/lisurvey.html.

[22] G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, and M. Scholl. RQL: A Declarative Query Language for RDF. In *The 11th Intl. World Wide Web Conference (WWW2002)*, Honolulu, Hawaii, USA, May 7–11 2002.

[23] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42(4):741–843, 1995.

[24] D. B. Lenat and R. V. Guha. *Building Large Knowledge-Based Systems: Representation and Inference in the CYC Project*. Addison-Wesley, 1990.

[25] A. Levy, A. Mendelzon, Y. Sagiv, and D. Srivastava. Answering Queries Using Views. In *PODS*, 1995.

[26] A. Levy, A. Rajaraman, and J. Ordille. Querying Heterogeneous Information Sources Using Source Descriptions. In *VLDB*, pages 251–262, 1996.

[27] P. J. Marrón and G. Lausen. On processing XML in LDAP. In *VLDB*, 2001.

[28] W. May. Mondial database. http://www.informatik.uni-freiburg.de/~may/Mondial, 2000.

[29] W. May. Querying linked XML document networks in the Web. In *The 11th Intl. World Wide Web Conference (WWW2002)*, Honolulu, Hawaii, USA, May 7–11 2002.

[30] E. Mena, V. Kashyap, A. Sheth, and A. Illarramendi. OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. *Distributed and Parallel Databases*, 8(2):223–271, 2000.

[31] L. Miller. RDF query using SquishQL. Available at http://swordfish.rdfweb.org/rdfquery/, 2001.

[32] T. Millstein, A. Levy, and M. Friedman. Query containment for data integration systems. In *PODS*, 2000.

[33] A. Seaborne. RDQL: A data oriented query language for RDF models. Available at http://www.hpl.hp.com/semweb/rdql.html, 2001.

[34] M. Sintek and S. Decker. RDF query and transformation language. Available at http://www.dfki.uni-kl.de/frodo/triple/, 2001.

[35] M. Wahl, T. Howes, and S. Kille. Lightweight directory access protocol (v3). RFC 2251, Dec. 1997.

OntoShare: Using Ontologies for Knowledge Sharing

John Davies, Alistair Duke and
Audrius Stonkus
BTexact Technologies, Orion 5/12,
Adastral Park,
Ipswich IP5 3RE, UK
john.nj.davies@bt.com
http://www.labs.bt.com/people/daviesn2/

ABSTRACT

An ontology-based knowledge sharing system OntoShare is described. RDF(S) and RDF are used to specify and populate an ontology, based on information shared between users in virtual communities. We begin by discussing the advantages that use of Semantic Web technology afford in the area of knowledge management tools. The way in which OntoShare supports WWW-based communities of practice is described. Usage of OntoShare semi-automatically builds an RDF-annotated information resource for the community (an potentially for others also). Observing that in practice the meanings of and relationships between concepts evolve over time, OntoShare supports a degree of ontology evolution based on usage of the system – that is, based on the kinds of information users are sharing and the concepts (ontological classes) to which they assign this information. We conclude by describing some avenues of ongoing and future research and a planned evaluation exercise.

1. INTRODUCTION

There are now more than two billion documents in the WWW, which are used by more than 300 million users globally, and millions more pages on corporate intranets. The continued rapid growth in information volume makes it increasingly difficult to find, organise, access and maintain the information required by users. Tim Berners-Lee and others [1] have proposed a semantic web that provides enhanced information access based on the exploitation of machine-processable metadata. We are particularly interested in the new possibilities afforded by semantic web technology in the area of knowledge management and we discuss this below before moving on in the rest of the paper to describe OntoShare, a system for supporting Semantic Web-based communities of practice.

Central to the vision of the Semantic Web are ontologies. Ontologies are seen as facilitating knowledge sharing and re-use between agents, be they human or artificial [2]. They offer this capability by providing a consensual and formal conceptualisation of a given domain. As such, the use of ontologies and supporting tools offer an opportunity to significantly improve knowledge management capabilities in large organisations and it is their use in this particular area that is the subject of this paper. In OntoShare, an ontology specifies a hierarchy of concepts (ontological classes) to which users can assign information. In this process, important metadata is extracted and associated with the community information resource using RDF annotations.

1.1 The Semantic Web and Knowledge Management

Due to a number of factors, including globalisation and the impact of the Internet, many organisations are increasingly geographically dispersed and organised around virtual teams. As noted in, for example, [3], such organisations need knowledge management and organisational memory tools that encourage users to understand each other's changing contextual knowledge and foster collaboration while capturing, representing and interpreting the knowledge resources of their organisations.

Important information is often scattered across Web and/or intranet resources. Traditional search engines return ranked retrieval lists that offer little or no information on the semantic relationships among documents. Knowledge workers spend a substantial amount of their time browsing and reading to find out how documents are related to one another and where each falls into the overall structure of the problem domain. Yet only when knowledge workers begin to locate the similarities and differences among pieces of information do they move into an essential part of their work: building relationships to create new knowledge.

So information retrieval traditionally focuses on the relationship between a given query (or user profile) and the information store. On the other hand, exploitation of interrelationships between selected pieces of information (which can be facilitated by the use of ontologies) can put otherwise isolated information into a meaningful context. The implicit structures so revealed help users use and manage information more efficiently [4].

Knowledge management tools are needed that integrate the resources dispersed across web resources into a coherent corpus of interrelated information. Previous research in information integration (see for example [5]) has largely focused on integrating heterogeneous databases and knowledge bases, which represent information in a highly structured way, often by means

of formal languages. In contrast, the Web consists to a large extent of unstructured or semi-structured natural language texts.

Ontologies offer an alternative way to cope with heterogeneous representations of Web resources. The domain model implicit in an ontology can be taken as a unifying structure for giving information a common representation and semantics.

1.2 Communities of Practice & the Semantic Web

The notion of *communities of practice* [6] has attracted much attention in the field of knowledge management. Communities of practice are groups within (or sometimes across) organisations who share a common set of information needs or problems. They are typically not a formal organisational unit but an informal network, each sharing in part a common agenda and shared interests or issues. In one example it was found that a lot of knowledge sharing among copier engineers took place through informal exchanges, often around a water cooler. As well as local, geographically based communities, trends towards flexible working and globalisation has led to interest in supporting dispersed communities using Internet technology [7]. The challenge for organisations is to support such communities and make them effective. Provided with an ontology meeting the needs of a particular community of practice, knowledge management tools can arrange knowledge assets into the predefined conceptual classes of the ontology, allowing more natural and intuitive access to knowledge.

Knowledge management tools must give users the ability to organize information into a controllable asset. Building an intranet-based store of information is not sufficient for knowledge management; the relationships within the stored information are vital. These relationships cover such diverse issues as relative importance, context, sequence, significance, causality and association. The potential for knowledge management tools is vast; not only can they make better use of the raw information already available, but they can sift, abstract and help to share new information, and present it to users in new and compelling ways

In this paper, we describe the OntoShare system which facilitates and encourages the sharing of information between communities of practice within (or perhaps across) organizations and which encourages people – who may not previously have known of each other's existence in a large organization – to make contact where there are mutual concerns or interests. As users contribute information to the community, a knowledge resource annotated with metadata is created. Ontologies are defined using RDF Schema (RDFS) and populated using the Resource Description Framework (RDF). (RDF [20] is a W3C recommendation for the formulation of metadata for WWW resources. RDF(S) [21] extends this standard with the means to specify domain vocabulary and object structures – that is, concepts and the relationships that hold between them).

In the next section, we describe in detail the way in which OntoShare can be used to share and retrieve knowledge and how that knowledge is represented in an RDF-based ontology. We then proceed to discuss in Section 3 how the ontologies in OntoShare evolve over time based on user interaction with the system and motivate our approach to user-based creation of RDF-annotated information resources.

2. SHARING AND RETRIEVING KNOWLEDGE IN ONTOSHARE

OntoShare is an ontology-based WWW knowledge sharing environment for a community of practice that models the interests of each user in the form of a user profile. In OntoShare, user profiles are a set of topics or ontological concepts (classes declared in RDFS) in which the user has expressed an interest. OntoShare has the capability to summarize and extract key words from WWW pages and other sources of information shared by a user and it then shares this information with other users in the community of practice whose profiles predict interest in the information.

OntoShare is used to store, retrieve, summarize and inform other users about information considered in some sense valuable by an OntoShare user. This information may be from a number of sources: it can be a note typed by the user him/herself; it can be an intra/Internet page; or it can be copied from another application on the user's computer.

As we will see below, OntoShare also modifies a user's profile based on their usage of the system, seeking to refine the profile to better model the user's interests.

2.1 Sharing Knowledge in OntoShare

When a user finds information of sufficient interest to be shared with their community of practice, a 'share' request is sent to OntoShare via the Java client that forms the interface to the system. OntoShare then invites the user to supply an annotation to be stored with the information. Typically, this might be the reason the information was shared or a comment on the information and can be very useful for other users in deciding which information retrieved from the OntoShare store to access. At this point, the system will also match the content being shared against the concepts (ontological classes) in the community's ontology. Each ontological class is characterized by a set of terms (keywords and phrases) and the shared information is matched against each concept using the vector cosine ranking algorithm [11]. The system then suggests to the sharer a set of concepts to which the information could be assigned. The user is then able to accept the system recommendation or to modify it by suggesting alternative or additional concepts to which the document should be assigned.

When information is shared in this way, OntoShare performs four tasks:

i. an abridgement of the information is created, to be held on the user's local OntoShare server. This summary is created using the ViewSum text summarization tool. The summarizer extracts key theme sentences from the document. It is based on the frequency of words and phrases within a

document, using a technique based on lexical cohesion analysis [22]. Access to this locally held summary enables a user to quickly assess the content of a page from a local store before deciding whether to retrieve the (larger amount of) remote information.

ii. the content of the page is analyzed and matched against every user's profile in the community of practice. As when recommending concepts to the user, the vector cosine ranking model is used: here, however, the shared information is matched against the set of terms (words and phrases) created from the union of all terms associated with the concepts to which has user has subscribed (i.e. the concepts which make up the user profile). If the profile and document match strongly enough, OntoShare emails the user, informing him or her of the page that has been shared, by whom and any annotation added by the sharer.

iii. the information is also matched against the sharer's own profile in the same way. If the profile does not match the information being shared, the system will suggest one or more concepts which strongly match the shared information that the user can then add to their profile. Thus OntoShare has the capability to adaptively learn users' interests by observing user behaviour.

iv. for each document shared, an instance of the class *Document* is created, with properties holding meatadata including keywords, an abridgement of the document, document title, user annotation, universal resource locator (URL), the sharer's name and date of storage. (The ontological structure of the OntoShare store is described in detail in the next section)

In this way, a shared and enhanced information resource is built up in the OntoShare store based on user contributions. Given that users must make a conscious decision to store information, the quality of the information in the OntoShare store is high - it is effectively pre-filtered by OntoShare users. Thus each user leverages the assessment of the information made by all the other users.

2.2 Ontological Representation

We said above that each piece of shared information leads to the creation of a new entry in the OntoShare store and that this store is effectively an ontology represented in RDF(S) and RDF. We now set this out in more detail. RDFS is used to specify the classes in the ontology and their properties. RDF is then used to populate this ontology with instances as information is shared. Figure 1 shows a slightly simplified version of the ontology for a community sharing information about the Semantic Web, along with an example of a single shared document ("Document_1").

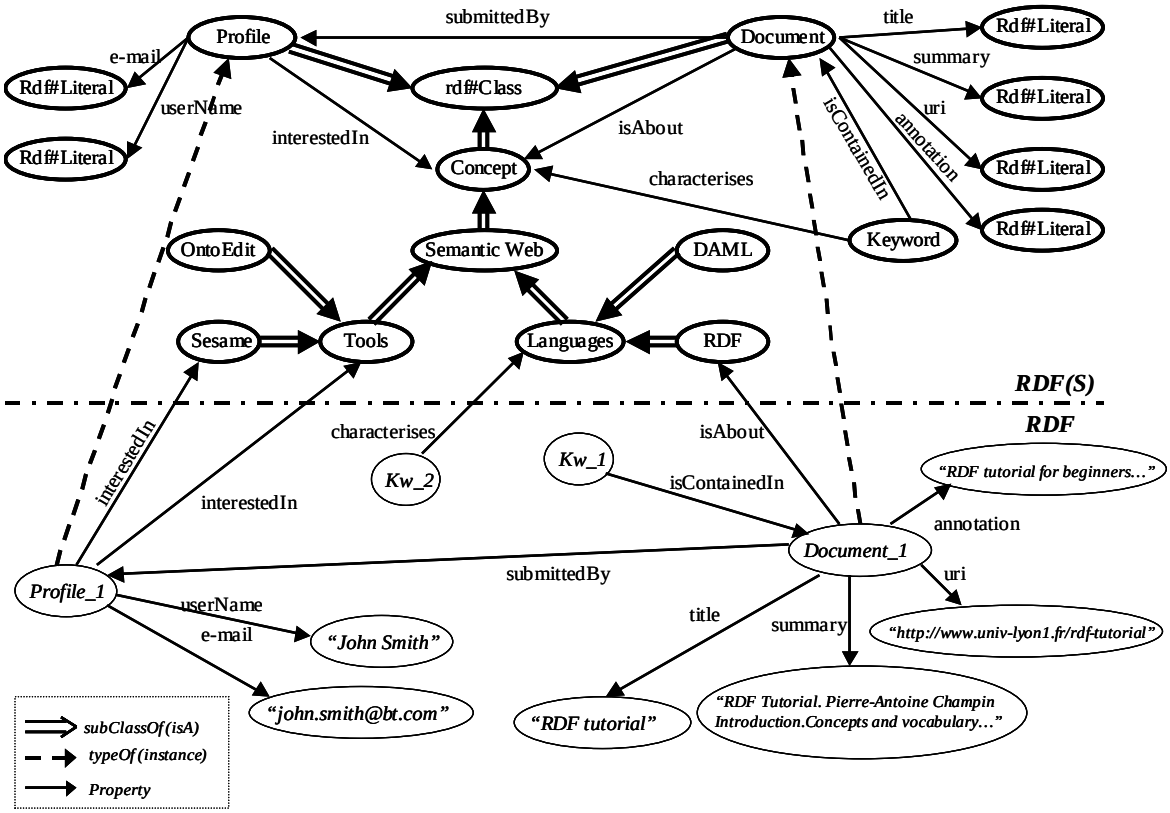

Figure 1**. Ontological Structure in OntoShare**

It is nor our intention to describe each class and property and their function here but we will mention a few key aspects. Firstly, notice *Concept* and its subclasses: this is the set of concepts which the community of practice at hand is interested in. Note that in the current version of OntoShare, the concept structure is limited to a strict hierarchy. Another key class is *Document*, which is the class used to represent shared information: each document shared generates an instance of *Document* with the set of properties shown. *Document_1*, for example, was stored by John Smith into the concept *RDF* with the annotation "RDF tutorial for beginners…" with the summary and URI as shown in Figure 1. It also has a set of keywords associated with it. (For simplicity, note that here we show only one keyword *Kw_1*, which is an instance of the class *Keyword*, as is *Kw_2* and furthermore that the instance (typeOf) relation is not shown for these keywords, nor is the fact that Keyword is a subclass of rdfs#Resource). The third central class is *Profile*, instances of which represent user information, including the concepts in which they are interested, their names and email addresses. *Profile_1*, for example, is the profile of a user with name "John Smith". Finally, note that keyword Kw_2 is one of (possibly many) terms (words and phrases) which characterize the concept *Language*.

Below we include excerpts from the RDFS and RDF (in XML notation) used to represent the ontology depicted above. We see the declarations of the classes *Document*, *Profile* and *Keyword* in RDF(S), followed by the descriptions of *Document_1* and the user profile of John Smith in RDF.

```
<?xml version="1.0" encoding="UTF-8" ?>
<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:ontoshare="http://www.bt.com/ontoshare#">


<!--*************RDFS SCHEMA ************* -->
    <rdfs:Class rdf:ID="Document" />
    <rdfs:Class rdf:ID="Profile" />
    <rdfs:Class rdf:ID="Keyword" />

<!--  Document properties -->
<rdf:Property rdf:ID="submitted_by">
<rdfs:domain rdf:resource="#Document" />
<rdfs:range rdf:resource="#Profile" />
</rdf:Property>
        ..............
        ..............



<!--  ************ RDF DATA ************  -->

<!--  DOCUMENTS -->
<Document rdf:ID="Document_1">
<title>RDF Tutorial</title>
<uri>http://www710.univ-lyon1.fr/champin/rdf-
tutorial</uri>
<submitted_by>#Profile_1</submitted_by>
<summary>doc summary goes here</summary>
```

```
<isAbout rdf:resource="#RDF" ontoshare:ID="7" />
<annotation>RDF tutorial for
beginners...</annotation>
</Document>

<!--  PROFILES -->
<Profile rdf:ID="Profile_1">
<user_name>John Smith</user_name>
<email>john.smith@bt.com</email>
<interestedIn rdf:resource="#Sesame"
ontoshare:ID="5" />
<interestedIn rdf:resource="#Tools"
ontoshare:ID="2" />
    </Profile>
        ……
        ……
```

2.3 Retrieving *explicit* knowledge in OntoShare

In this section, we discuss the ways in which OntoShare facilitates access to and the automatic sharing of the information shared by users.

- **Email notification**

As described above, when information is shared in OntoShare, the system checks the profiles of other users in the community of which the user is a member. If the information matches a user's profile sufficiently strongly, an email message is automatically generated and sent to the user concerned, informing the user of the discovery of the information. Thus in cases where a user's profile indicates that they would have a strong interest in information shared, they are immediately and proactively informed about the appearance of the information.

- **Searching the community store – accessing information and people**

Via button on their OntoShare home page, a user can supply a query in the form of a set of key words and phrases in the way familiar from WWW search engines. OntoShare then retrieves the most closely matching pages held in the OntoShare store, using a vector space matching and scoring algorithm [11].

The system then displays a ranked list of links to the pages retrieved and their abridgements, along with the scores of each retrieved page and any annotation made by the original sharer is also shown. Importantly, the user can elect to simultaneously search for other users by selecting the appropriate check box. We will have more to say about this capability to identify other *users* as well as *information* in section 4 when we look at accessing *tacit* knowledge via other users using OntoShare.

- **Personalised Information**

A user can also ask OntoShare to display "Documents for me" as shown in the top right pane of Figure 2 below. The system then interrogates the OntoShare store and retrieves the most recently stored information. It determines which of these pages best match the user's profile. The user is then presented with a list of links to the most recently shared information, along with a summary, annotations where provided, date of storage, the sharer and an indication of how well the information matches the user's profile (the thermometer-style icon in Figure 2 below).

In addition, 2 buttons are provided (on the button bar at the bottom of the screen in Figure 2) so that the user can indicate interest or disinterest in a particular piece of information – this feedback will be used to modify the user's profile. At this point, the system will match the content of the current document against each concept (ontological class) in the community's ontology. As described above, each ontological class is characterized by a set of terms (keywords and phrases) and the shared information is matched against he term set of each concept using the vector cosine ranking algorithm [8]. The system then identifies the set of zero or more concepts that match the information above a given ranking threshold and suggests to the sharer that this set of concepts be added to or removed from their profile in the cases of user interest or disinterest respectively. The user is then free to accept the system recommendation or to modify it by selecting from the set of suggested concepts.

Two further operations are possible on documents presented to the user. These operations are selected from the "Documents" menu. Firstly, a user can add their own annotation to information stored by another user. Secondly, a user can request that OntoShare identifies other users with an interest in the information under consideration.

This "Documents for me" information is in fact displayed on the user's OntoShare home page, so that whenever they access the system, they are shown the latest information. Figure 2 is a typical OntoShare home page.

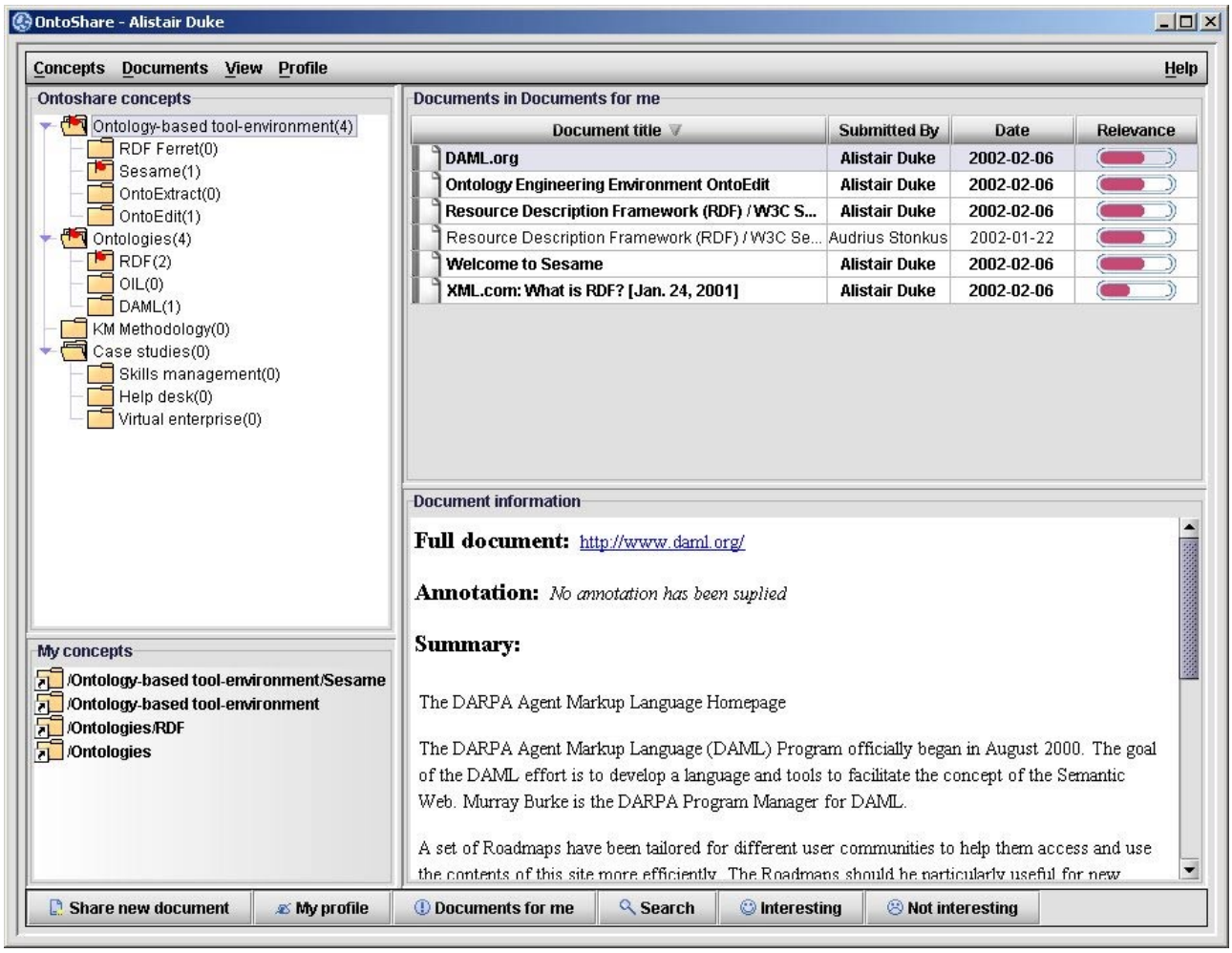

Figure 2**. Typical OntoShare Home Page**

3. CREATING EVOLVING ONTOLOGIES

In section 2, we described how, when a user shares some information, the system will match the content being shared against each concept (class) in the community's ontology. Recall that each ontological class is characterized by a set of terms (keywords and phrases) and that following the matching process, the system suggests to the sharer a set of concepts to which the information could be assigned. The user is then able to accept the system recommendation or to modify it by suggesting alternative concept(s) to which the document should be assigned. It is at this point that an opportunity for ontology evolution arises.

Should the user indeed override the system's recommended classification of the information being shared, the system will attempt to modify the ontology to better reflect the user's conceptualisation, as follows. The system will extract the keywords and keyphrases from the information using the ViewSum system mentioned above. The set of such words and phrases are then presented to the user as candidate terms to represent the class to which the user has assigned the information. The user is free to select zero or more terms from this list and/or type in words and phrases of his own. The set of terms so identified is then added to the set of terms associated with the given concept, thus modifying its characterization.

We call this approach usage-based ontology evolution and in this way the characterization of a given concept evolves over time, this evolution being based on input from the community of users. We believe that this ability to change as users' own conceptualization of the given domain changes is a powerful feature which allows the system to better model the consensual ontology of the community. Clearly, this level of evolution is limited to changing the semantic characterization of ontological

classes and does not support, for example, the automatic suggestion of new classes to be added to the ontology. More advanced ontology evolution is the subject of ongoing research and is described briefly in Section 5. It is also worh noting that we have not concerned ourselves with ontology versioning (tracking and managing changes to an ontology) here. This is an important issue with regard to ontology evolution and the reader is referred to, for example, [26], [27] for details of work in this area.

As well as usage-based evolution, we have seen above how users also indirectly annotate the information as a side-effect of sharing it with the community and we discuss and motivate this approach below.

Pragmatically speaking, it is the case at the time of writing that only a very small proportion of WWW- and intranet-based information resources are annotated with RDF (meta)data. It is therefore beneficial to provide a system wherein such annotation effectively occurs as a side-effect of normal usage.

Another important observation is that it is in the general case impossible to cover the content of a document exhaustively by an RDF description. In practice, RDF descriptions can never replace the original document's content: any given RDF description of a set of resources will inevitably give one particular perspective on the information described. Essentially, a metadata description can never be complete since all possible uses for or perspectives on data can never enumerated in advance.

Our approach accommodates this observation however in the sense that each community will create its own set of metadata according to its own interest in and perception of information that is added to its store. It is very possible that the same information could be shared in two separate communities and emerge with different metadata annotations in each.

4. EXPERTISE LOCATION AND TACIT KNOWLEDGE

In section 2, we focused on the *technical* aspects of OntoShare and on the sharing and storing of explicit knowledge. Explicit knowledge we take to be that knowledge which has been codified in some way. This codification can take place in many different media (paper, WWW page, audio, video, and so on). In the context of OntoShare, by explicit knowledge, we mean the information shared in OntoShare, along with the meta-information associated with it such as the sharer, the annotations attached to it, and so forth. We now turn to the *social* aspects of the system and tacit knowledge.

A large amount of the knowledge within an organization may of course not be codified: it may be personal, context-specific and difficult to write down, and may be better transmitted through a master-apprentice "learning by watching and copying" arrangement. Such knowledge is referred to as *tacit* knowledge [9]. When tacit knowledge is difficult to make explicit (codify), we need to find new ways of transmitting the knowledge through an organization. Failure to do so can lead to loss of expertise when people leave, failure to benefit from the experience of others, needless duplication of a learning process, and so on.

One way in which a system such as OntoShare can encourage the sharing of tacit knowledge is by using its knowledge of the users within a community of practice to put people who would benefit from sharing their (tacit) knowledge in touch with one another automatically.

One important way we gain new insights into problems is through 'weak ties', or informal contacts with other people [10, 11]. Everyone is connected to other people in social networks, made up of stronger or weaker ties. Stronger ties occur between close friends or parts of an organization where contact is maintained constantly. Weak ties are those contacts typified by a 'friend of a friend' contact, where a relationship is far more casual. Studies have shown that valuable knowledge is gathered through these weak ties, even over an anonymous medium such as electronic mail and that weak ties are crucial to the flow of knowledge through large organizations. People and projects connected to others through weak ties are more likely to succeed than those not [12, 13].

User profiles can be used by the OntoShare system to enable people to find other users with similar interests. The user can request OntoShare to show them a list of people with similar interests to themselves. OntoShare then compares their profile with that of every user in the store and a list of names of users whose interests closely match their own. Each name is represented as a hypertext link which when clicked initiates an email message to the named user. Recall that profiles in OntoShare are a set of phrases and thus the vector space model can be used to measure the similarity between two users. A threshold can then be used to determine which users are of sufficient similarity to be deemed to 'match'.

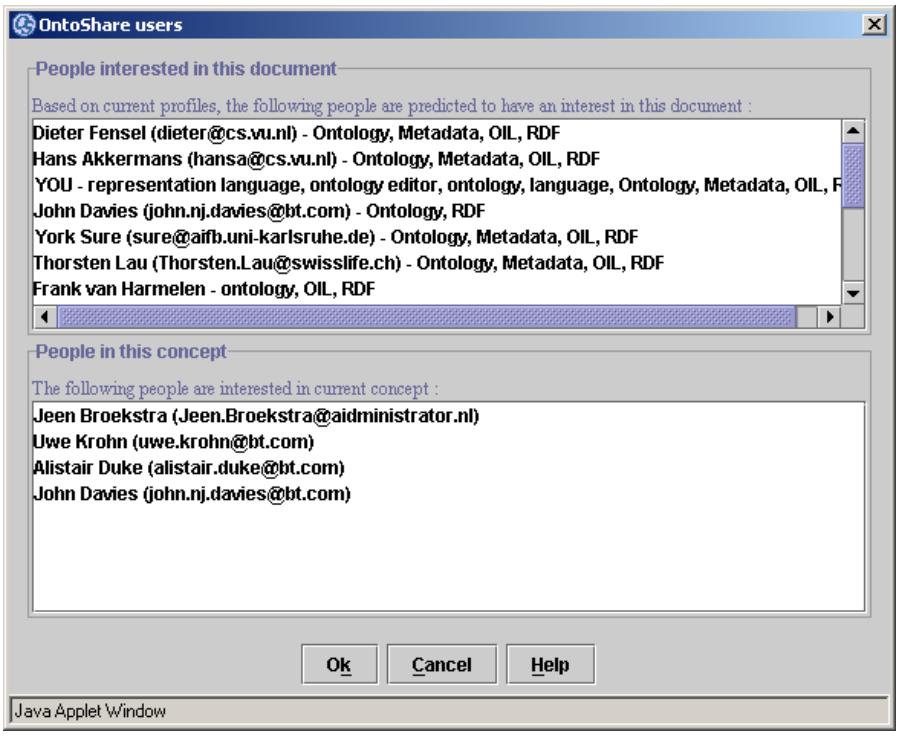

Figure 3. Identifying expertise on OntoShare.

This notion is extended to allow a user to view a set of users who are interested in a given document. OntoShare determines which members of the community 'match' the relevant document above a predetermined threshold figure and presents back to the user a list of user names. As before, these names are presented as hypertext links, allowing the user to initiate an email message to any or all of the users who match the document. Figure 3 shows typical output from this process.

In addition, as already mentioned in section 2.3, a user can carry out a keyword search on other users and thus identify users with an interest in a particular subject.

In this way, OntoShare, while not claiming to actually capture tacit knowledge, provides an environment which actively encourages the sharing of tacit knowledge, perhaps by people who previously would not otherwise have been aware of each other's existence.

5. EVALUATION

OntoShare is a recently developed system and no formal evaluations have yet taken place. We briefly describe here an evaluation due to start in April 2002. The user group for the study will consist of approximately 30 researchers, developers and technical marketing professionals from the research and development arm of a large telecommunications firm. The interests of the users fall into 3 main groupings: conferencing, knowledge and information management and personalization technologies. It is felt that three separate yet overlapping topic areas will constitute an interesting mix of interests for the purposes of the trial.

The case study will commence with a workshop involving the practitioners in order to develop an ontology that encompasses the research fields with particular emphasis upon the overlap between them. OntoEdit [17] will be used to create the ontology for the research areas. This will then be uploaded to SESAME [18], allowing it to be viewed used as the ontology in OntoShare (which contains a module for reading ontological information from SESAME) and provide access to the ontology for other ontology tools with a similar capability. The ontology will automatically evolve and extend over the course of the study as documents are added to OntoShare. The effectiveness of this evolutionary process will be considered in the evaluation exercise. Qualitative and quantitative measures of the trial are being devised. The main evaluation criterion is to what degree the application of tools and methodology can ensure that knowledge discovered by individuals can be transferred to the most appropriate members of the user group. An interesting secondary outcome we wish to look at is the extent to which the ontology built up by the community is useful to other users in other contexts. In this regard, we plan to offer a searching and browsing facility over the community's information using the QuizRDF system [23] for other users outside the community.

6. FURTHER & RELATED WORK

Research and development of OntoShare is ongoing. A particular area of focus currently is the ontological structure: a strict hierarchy (taxonomy) of concepts about which the communities wants to represent and reason may prove ultimately limiting and various possibilities for allowing a more expressive concept map are under consideration. One such is that OntoShare will be developed beyond the subclass/superclass concept hierarchy with *IsRelatedTo* properties, allowing "horizontal" links between concepts. The exploitation of this additional information is again matter for further research. One proposal is that when seeking to match users to other users, the system can use some notion of tree-matching, taking into account the concepts in the users' profiles as well as not only the *IsA* (*subClassOf*) links but also the *IsRelatedTo* links. These richer ontologies may be better represented in a more expressive language such as OWL, the upcoming standard from the W3C Web Ontology working group [25].

A further research area is the automatic identification and incorporation of *new* concepts as they emerge in the community. Work on this is however at a very early stage and is beyond the scope of this paper.

Turning to related work, Staab et al. [28] describe a system for building and maintaining community web portals. As with OntoShare, a ontology-based is taken and an ontology is used to structure and access information, using F-Logic as its underlying language for ontology representation and querying. Relatively sophisticated querying is supported, offering a degree of inferencing in the query engine not offered in OntoShare. Semi-structured information provision is supported by the use of wrappers. User profiling and automatic alerting are not supported, neither is the ability to change the semantic characterization of a class as in OntoShare.

RiboWeb [29] is another example of an ontology-based community portal RiboWeb holds information about ribosome data and computational models for the processing thereof. Most data are scientific papers manually linked to the appropriate ontological categories. Knowledge engineers maintain the data and metadata, rather than the data being provided by the community itself as in OntoShare.

7. CONCLUDING REMARKS

We have described OntoShare, an ontology-based system for sharing information among users in a virtual community of practice. We motivated the use of Semantic Web technology for KM tools and described how ontologies in OntoShare are defined in RDFS. Communities are able to automatically share information and create RDF-annotated information resources as a side-effect of this activity. Furthermore, these information resources are then of course available to other RDF-based tools for processing: the community semi-automatically creates an ontology-based annotated information resource for use by itself and others.

Importantly, the ontology used by a given community in OntoShare can change over time based on the concepts represented and the information that users choose to associate with particular concepts. This is a significant advantage over a community attempting to reach consensus on a set of concepts and how they relate to another at the outset that is then difficult or impossible to change. Much remains to be done in this area however, particularly with regard to the introduction of new concepts. In addition, users have personal profiles according to the concepts in which they have declared an interest and these profiles also evolve automatically, seeking to match more closely a user's information needs and interests based on the usage they make of the system.

We indicated some further directions of research and briefly discussed an ongoing evaluation of the system. OntoShare exemplifies the much-improved knowledge management tools that the advent of the Semantic Web and its support for ontologies makes possible.

8. ACKNOWLEDGEMENTS

The research described in this project was funded in part by the EU IST project OnToKnowledge (www.ontoknowledge.org), reference IST-1999-10132.

9. REFERENCES

[1] T. Berners-Lee, J. Hendler, and O. Lassila: The Semantic Web. Scientific American (May 2001)

[2] D. Fensel: Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce. Springer-Verlag, Berlin (2001)

[3] Goldman-Segall, R. & Rao, S.V.: A collaborative online digital data tool for creating living narratives, in : Organisational Knowledge Systems, 31st Hawaii International Conference on Systems Science, Hawaii, USA (1998)

[4] Shipman, F.M., Marshall, C.C. & Moran, T.P.: Finding and using implicit structure in human-organized spatial layouts of information. CHI-95 (1995)

[5] Hearst, M. A.: Information integration. IEEE Intelligent Systems (Sept./Oct. 1998) 12-24

[6] Seely-Brown, J. and Duguid, P., Organisational Learning and Communities of Practice, Organisational Science, Vol 2(1) (1991)

[7] Davies, N.J., Supporting Virtual Communities of Practice, in "Industrial Knowledge Management", Roy, R. (ed), Springer-Verlag (2000)

[8] Harman, D., Ranking Algorithms, in Frakes, W. & Baeza-Yates, R. Information Retrieval, Prentice-Hall, New Jersey, USA (1992)

[9] Polyani, M., 1966, *The Tacit Dimension*, London: Routledge & Paul.

[10] Granovetter, M., 1974, *The Strength of Weak Ties*, American Journal of Sociology, 78, pp 1360-1380.

[11] Granovetter, M., 1982, *The Strength of Weak Ties: A Network Theory Revisited*, in 'Social Structure and Network Analysis', Marsden, P. and Nan, L. (Editors), Sage Publications, California.

[12] Constant, D., Sproull, L. and Kiesler, S., 1996, *The Kindness of Strangers: The Usefulness of Electronic Weak Ties for Technical Advice*, Organisation Science, Vol 7, Issue 2, P119-135, 1996.

[13] Hansen, M.T., 1997, *The Search-Transfer Problem: The Role of Weak Ties in Sharing Knowledge Across Organisation Subunits*, Working Paper, Harvard Business School.

[14] Nonaka, I., 1994, *A Dynamic Theory of Organisational Knolwedge Creation*, Organisation Science, 5, 1.

[15] http://www.deepwoods.com

[16] Maxwell, C., 2000, *The Future of Work – Understanding the Role of Technology*, BT Technology Journal, 18, 1, Kluwer, Netherlands.

[17] OntoEdit, http://ontoserver.aifb.uni-karlsruhe.de/ontoedit/

[18] Broekstra, J., A. Kampman, F. van Harmelen. *Sesame: An Architecture for Storing and Querying RDF Data and Schema Informatio*n. In [19].

[19] Fensel, D., J. Hendler, H. Lieberman, and W. Wahlster (eds.): *Semantic Web Technology*, MIT Press, Boston, USA, to appear.

[20] Lassila, O. and R. R. Swick. *Resource Description Framework (RDF): Model and Syntax Specification*. Recommendation, World Wide Web Consortium, February 1999. http://www.w3.org/TR/REC-rdf-syntax/

[21] Brickley, D. and R.V. Guha. *Resource Description Framework (RDF) Schema Specification 1.0*. Candidate recommendation, World Wide Web Consortium, March 2000. http://www.w3.org/TR/2000/CR-rdf-schema-20000327

[22] Ahmed, K. & Benbrahim, M., *Text Summarisation: the role of Lexical Cohesion Analysis*, New Review of Document and Text Management, Vol 1, 321-335, 1995.

[23] Davies, N.J., Krohn, U. & Weeks, R., *QuizRDF: Search Technology for the Semantic Web*, to appear in [24].

[24] Davies, N.J., Fensel, D. & van Harmelen, F. (eds), *Ontology-based Knowledge Management: exploiting the Semantic Web*, Wiley, London, UK, to appear Autumn 2002.

[25] http://www.w3.org/2001/sw/WebOnt/

[26] Heflin, J. & Hendler, J., Dynamic Ontologies on the Web, Proc. AAAI-2000, MIT Press, Menlo Park, USA, 2000.

[27] Klein, M. & Fensel, D., Ontology Versioning on the Semantic Web, Proc. SWWS-01, 2001.

[28] Staab, S., Angele, J., Decker, S., Erdmann, M., Hotho, A., Maedche, A., Schnurr, H-P., Studer, R. & Sure, Y., Ontology-based Community Web Portals, Proc. AAAI-2000, MIT Press, Menlo Park, USA, 2000.

[29] Altmann, R., Bada, M., Chai, X., Carillo, M.W., Chen, R. & Abernethy, N., RiboWeb: an Ontology-based System for Collaborative Molecular Biology, IEEE Intelligent Systems 15(5), 1999.

WebScripter: World-Wide Grass-roots Ontology Translation via Implicit End-User Alignment

Research Paper Category

Martin Frank, Pedro Szekely, Robert Neches, Baoshi Yan, Juan Lopez
Distributed Scalable Systems Division
Information Sciences Institute
University of Southern California
{frank,szekely,rneches,baoshi,juan}@isi.edu

ABSTRACT

Ontologies define hierarchies of classes and attributes; they are meta-data: data about data. XML Schema and RDF Schema are both (lightweight) ontology definition languages in that sense. In the "traditional" approach to ontology engineering, experts add new data by carefully analyzing others' ontologies and fitting their new concepts into the existing hierarchy. In the emerging "Semantic Web" approach to ontology engineering, ordinary users may not look at anyone's ontology before creating theirs – instead, they may simply define a new local schema from scratch that addresses their immediate needs, without worrying how their data may some day integrate with others'.

This paper describes an approach and implemented system for translating between the countless mini-ontologies that the Semantic Web approach yields. In this approach, ordinary users graphically align data from multiple sources in a simple spreadsheet-like view without having to know anything about ontologies or even taxonomies. The resulting web of equivalency statements can then be mined to help other users find related ontologies and data, and to automatically align the data with theirs.

Categories and Subject Descriptors

H.1.2 [**Information Systems**]: User/Machine Systems—*Human information processing*; H.3.3 [**Information Systems**]: Information Search and Retrieval—*Information filtering, Relevance feedback*; H.3.5 [**Information Systems**]: Online Information Services—*Data sharing, Web-based services*

General Terms

Collaborative filtering, recommender systems, social information filtering, ontology alignment, ontology translation

Keywords

Meta-data, DAML, RDF Schema, RDF, XML Schema

1. INTRODUCTION

Imagine that you work for an emergency preparedness agency and that you were just handed the job of constructing and maintaining a list of public health experts employed by U.S. universities.

Doing this manually on the (non-semantic) Web would be a monumental effort, both in terms of the initial effort and in the continuous effort to keep the list up to date. The only options are to either do the job completely manually in a text file or spreadsheet (quickly outdated), or to write wrapper software specific for each university's Web pages that extracts the experts (the wrappers constantly break as universities change their Web page designs).

Now let us presume that all universities list their personnel in a Semantic Web [10] format, such as RDF Schema [1]. This improves on the current sitation (because you don't have to work instance by instance but rather concept by concept) but your job is still rather monumental because the sources will likely use a myriad of different ontologies.

We have a vision and partial implementation addressing this problem by (a) making it easy for individual users to graphically align the attributes of two separate externally-defined concepts, and (b) making it easy to re-use the alignment work of others.

2. OVERVIEW

Figure 1 depicts a number of home pages, marked up with DAML information about the authors, located somewhere in the world. The DAML instances in these Web pages are organized according to one or more ontologies, such as an ISI ontology of people, a Stanford ontology of people, and a Karlsruhe ontology of people. The challenge is to produce a report incorporating all of that information with minimal effort.

At a high level, the WebScripter concept is that users extract content from apparently ordinary Web pages and paste that content into what looks like an ordinary spreadsheet (lower left corner of Figure 1).

What users implicitly do in WebScripter - without expending extra effort - is to build up an articulation ontology containing equivalency statements. For example, this artic-

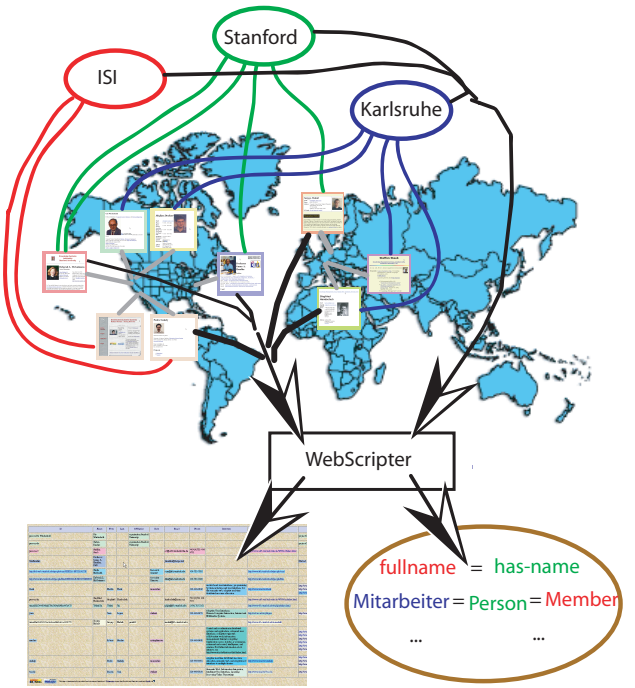

Figure 1: Working With Multiple Data Sources In Multiple Ontologies.

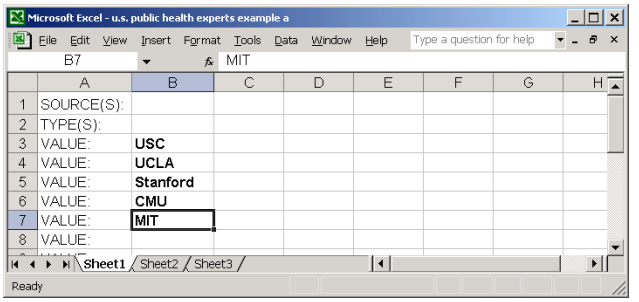

Figure 2: Visionary Example: The user types as-yet-unrecognized example values.

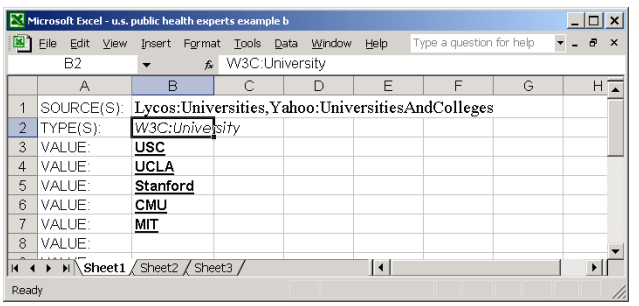

Figure 3: Visionary Example: The system determines data sources and a classification.

ulation ontology expresses that the attribute that ISI calls "fullname" is the same as the one Stanford calls "has-name"; and that the object Karlsruhe calls "Mitarbeiter" Stanford calls "Person" and ISI calls "Member" are the same for the purposes of this report (lower right corner of Figure 1).

We believe that in the long run, this articulation ontology will be more valuable than the data the users happened to obtain when they constructed the original report. Its equivalency information reduces the amount of work future WebScripter users have to perform when working in the same domain.[1] Thus, in some sense, you don't just use the Semantic Web when you use WebScripter, you help build it as you go along.

3. VISIONARY EXAMPLE

This section presents a detailed step-by-step vision of what WebScripter (and a future Semantic Web) could be; we will later present a step-by-step example of what our current implementation can already do (with existing RDF(S) data on the Web produced by others). In this example, the application is to quickly produce a self-updating list of faculty at U.S. universities that are public health experts, listing their specialization. The user starts WebScripter and types the names of several universities into the first column. At the point shown in Figure 2, truly nothing is known about these hand-typed values.

After the user selects "classify" from a menu, WebScripter uses a list of well-known indices to find an existing taxonomy that matches all of the typed phrases (note that commercial search engines do not have to be DAMLized to be useful to our reasoning here). Yahoo:UniversitiesAndColleges and Lycos:Universities both apply. The universities now appear underlined because they are recognized by the system - double-clicking on them brings up their web pages. The system then fetches their DAML-enabled Web pages in the background, and computes a minimal covering set of declared DAML IS-A types that cover all the universities. In our example, all of the current universities declare to be instances of the World-Wide Web Consortium's W3C:University concept (Figure 3).

The user now selects "find more" from the menu bar. The system will fetch every entity that the two known indices point to (several hundred). It simultaneously performs a different type of analysis: Which are the RDF(S) subclass-of types that are declared by more than 10% of the entities (result: U.N.:University, UsPostalService:Recipient, W3C:University, and IRS:NonProfitInstitution) [this is a recall test]? Of these, which apply to less than 1% of nearby categories of the same index (remaining result: W3C:University and U.N.:University) [this is a precision test]. The latter one is now automatically treated as an alternatively valid type, and WebScripter will include every entity declaring to be one of these types, thereby finding institutions not yet listed by the well-known indices. Note that there are no duplicate universities in this column (such as "UCLA" and "UC Los Angeles"). The challenge, of course, is to be able to determine that they are "different", as they subscribe to different DAML ontologies. One possiblity is that any Semantic Web description of an entity existing on the Web contain its normalized HTTP URL in a standardized attribute, which can serve as a simple unique id for comparisons across ontologies (first choice for disambiguation in our example). Another possibility is that they contain

[1]Who benefits depends on your willingness to share that information, of course - it could be the world, your organization, your workgroup, or just yourself.

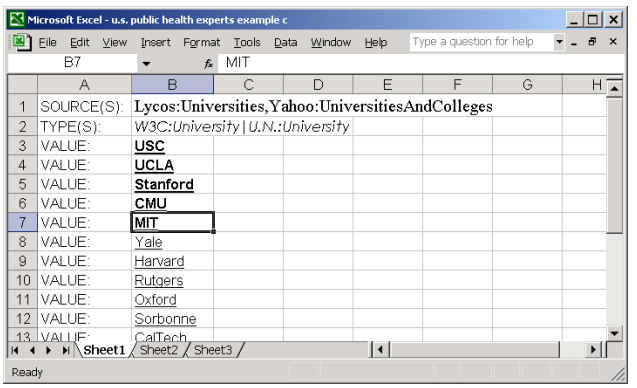

Figure 4: Visionary Example: The system autocompletes the user-provided values.

Figure 5: Visionary Example: Combining HTML navigation with embedded DAML semantics.

(possibly composite) keys that point into popular external ontologies, for the same reason ("companies in this ontology are uniquely identified by their UsTreas:IRS:TaxPayerId"), ("universities are identical if they point to the same UsPostalService:UsStreetAddress"). This gets the user to the state of Figure 4.

In this vision of a future Semantic Web, the user has to know little to get a lot of leverage out of the existing semantic information: (1) The user did not have to do anything but type out some university names that came to mind – he or she didn't have to understand an ontological query language or the notion of an ontology or even a taxonomy for that matter - yet the result is perfectly ontologically typed. (2) Very little DAML has to be in place for this to work: for this particular example, two external DAML ontologies of existing non-DAML university web sites should be sufficient for the inferencing of this example. (3) Data from two different ontologies can be seamlessly integrated without the need for pre-merging/translation between the ontologies.

In this example, the user now demonstrates that she wants to extract the nationality of the universities, in the following manner (Figure 5). She double-clicks on USC, which brings up a Web browser to the (hypothetically) DAML-enabled USC home page. The user then clicks on "Maps & Directions", and copies and pastes "United States" from that page, which is not just plain text but carries its embedded DAML type.[2]

In response, the system now fills in all those cells that use the same underlying W3C university definition, by inferring the ontological path from university to country and applying it to all other instances of this ontology. In our particular case, the user is best served by now doing the same for the UN-based university entry "Stanford" (not shown) because there are only two ontologies involved.[3] As a result, all missing countries in the second column are now filled in. The user selects a United States cell in the second column and invokes "filter by" from the right-click menu, checks "United States", and clicks OK, which removes Oxford and all other foreign-university rows. Performing a number of substantially similar steps, the user can navigate to the universities' chemistry, biology, and medical departments, from the department to the faculty, from the faculty to their research interests, and filter by a particular research area, resulting in the table shown in Figure 6. (As before, bold entries were provided or demonstrated by the user; but we are no longer underlining recognized cells below for readability).

In the end, what users want is a report containing the information that serves their immediate needs. In our approach, users build a report in steps, by manipulating the data it contains so far to refine it and to add more. This is a qualitatively easier task than working with a query, which is an inherently more abstract specification. In our approach, a final report may contain dozens or hundreds of single-step scripts that operate on DAML markup. The equivalent query could be enormously complicated (perhaps several pages long), but users never have to see it with this approach.

Now that this hypothetical WebScripter report is defined, its data can be refreshed at any time, and it itself can become the source for further Web scripting as it carries all its DAML within the generated HTML report.

4. IMPLEMENTED EXAMPLE

In our initial implementation we have focused on making it easy for ordinary (non-programming, never heard of ontologies) users to contruct reports from multi-ontology DAML data. This section first describes a step-by-step walkthrough of using WebScripter as implemented to combine DAML personnel data from different organizations on the Web. It then describes how the resulting implicit ontology alignment data benefits other users in constructing similar reports.

4.1 Constructing a first report from scratch

Imagine that you work for the government DAML program office, and that your job is to maintain a list of personnel funded by that program, and let's assume that all of the contractors provide their personnel data in some DAML format. The first task is to find the URLs where the vari-

[2]Note that one would not have to internally instrument a Web browser to achieve this level of integration – one could know which page the user is looking at through a proxy Web server and receive the copied HTML+DAML out of the window system's paste buffer.

[3]If there is more than two WebScripter could attempt to produce a generalized "fuzzy" script that will work for all remaining university ontologies given two (or more) examples ("extract the attribute whose name contains Country or Nation in the top-level concept or in a sub-concept called Location or Address").

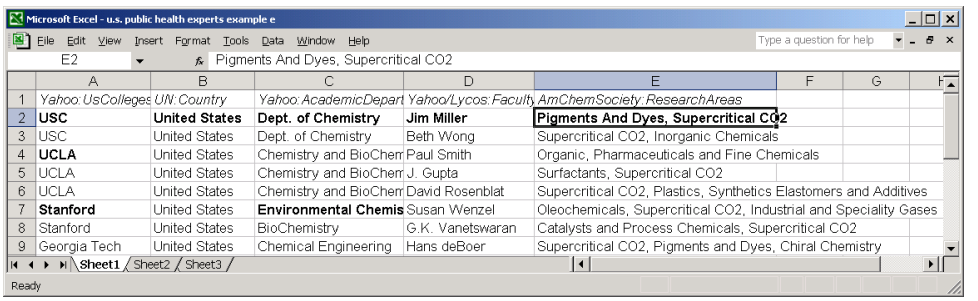

Figure 6: Visionary Example: The end result in this fictious example.

ous DAML resides. BBN's crawled ontology library comes closest to a Yahoo-style portal for DAML content [2]. This site contains a registry for DAML content root files, which a crawler uses as starting points to find more DAML files. Teknowledge built a DAML search engine for that ontology library which is a good starting place for finding DAML content [3].

In this example, the DAML sources can be found by querying the Teknowledge search engine for the terms "Person", "Employee", and "Staff" (which will return a large number of hits of non-DAML contractors), or alternatively it can be found by collecting the regular project Web pages and personal home pages of the DAML contractors (because they embed DAML content inside the HTML pages).

For the sake of this example, we started WebScripter and loaded DAML from just the Stanford Database and Knowledge Systems groups by copying and pasting the URLs of their DAML pages into WebScripter's "Add DAML" dialog box. WebScripter then displays the class hierarchy of that DAML, intermixing the concepts from the two separate ontologies. The user can browse the content by selecting classes, which displays all of their (local and inherited) attributes as columns, and their data instances as rows.

In this example, we started a new report by (1) choosing "New Report" from a menu, (2) selecting Person in the class hierarchy, and (3) selecting three columns of Person to include in the report. The latter is done by selecting a cell in the data display for Person and choosing "Add as new column" from the right-click menu, once each for the Has-Full-Name, Has-Phone-Number, and Has-Email-Address columns. The resulting WebScripter display is shown in Figure 7. (Note that the first of the four columns, the DAML instance identifier column, was automatically inserted when the first column was added to the report. The column is hidden from the generated report Web page by default.)

In this example, we will now add and align data from a different research group using a different ontology. This is done by (1) selecting PhDStudent in the class hierarchy to display its instance data, (2) selecting a cell in the "name" column of that instance data and choosing "Add to column 1" from the right click menu, and (3) repeating the second step for the "phone" and "email" columns. Figure 8 shows the combined data from the two groups.

This in a nutshell is how WebScripter looks to the users. This report can then be published in various formats, including as a plain Web page that color-codes its content based on where it came from; Figure 9 shows a snapshot of a large DAML personnel report that loads data from more than 30

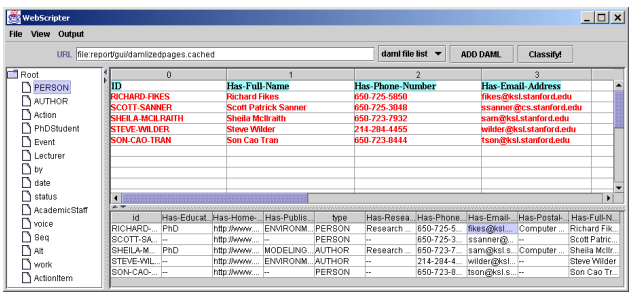

Figure 7: Implemented Example: Initial report of Stanford KSL personnel.

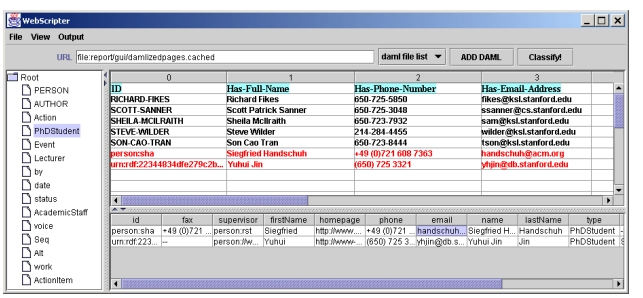

Figure 8: Implemented Example: Adding and aligning Stanford Database personnel.

different sources [4]. The largest such report we have generated is 3.3MB, taking 8.7MB of DAML input, and running for about 45 seconds.

The Web page embeds the WebScripter report definition, thus it can be re-run at any time and will then possibly show more people (presuming their DAMLized Web pages can be found by following just one link from the two group Web pages, and presuming their DAML instance data follows one of the two ontologies).

There are a large number of WebScripter features that we will not discuss here – such as un-loading DAML sources, deleting columns, re-arranging columns, filtering rows, and sorting by multiple criteria, and so on – because they are what you would expect from any DAML report generator. Instead, we'll focus on the generated DAML equivalency statements shown in Table 1.

These statements can be automatically published on a Web site and registered as a new DAML content root in BBN's DAML content library. Consequently, you can then

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdf="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:daml="http://www.daml.org/2001/03/daml+oil#">

<rdfs:Class
  rdf:about="http://ksl.stanford.edu/Projects/DAML/ksl-daml-desc.daml#PERSON">
  <daml:sameClassAs rdf:resource=
  "http://www.semanticweb.org/ontologies/swrc-onto-2000-09-10.daml#PhDStudent"/>
</rdfs:Class>

<rdfs:Property
  rdf:about="http://ksl.stanford.edu/Projects/DAML/ksl-daml-desc.daml#Has-Full-Name>
  <daml:samePropertyAs rdf:resource=
  "http://www.semanticweb.org/ontologies/swrc-onto-2000-09-10.daml#name"/>
</rdfs:Property>
<rdfs:Property
  rdf:about="http://ksl.stanford.edu/Projects/DAML/ksl-daml-desc.daml#Has-Phone-Number>
  <daml:samePropertyAs rdf:resource=
  "http://www.semanticweb.org/ontologies/swrc-onto-2000-09-10.daml#phone"/>
</rdfs:Property>
<rdfs:Property
  rdf:about="http://ksl.stanford.edu/Projects/DAML/ksl-daml-desc.daml#Has-Email-Address>
  <daml:samePropertyAs rdf:resource=
  "http://www.semanticweb.org/ontologies/swrc-onto-2000-09-10.daml#email"/>
</rdfs:Property>

</rdf:RDF>
```

Table 1: Implemented Example: Resulting DAML equivalency statements.

make use of the equivalency statements by selecting the "Extended with Equivalence" option in Teknowledge's DAML search engine (note that it can take up to 24 hours for the statements to make it into BBN's cache and then up to another week from there into Teknowledge's search engine cache). Concretely, if you for example now query for all instances of person ("?x type Person") in the first ontology in that fashion you will now also retrieve PhDStudent instances from the second ontology.

4.2 Constructing a second report using the alignment data

We have also implemented an intial use of the WebScripter-generated equivalency statements in WebScripter itself: if you start it with the *insert-equivalents* flag it will automatically add and align any classes that it has *sameClassAs* and *sameInstanceAs* data for. It reads these equivalency statements from a fixed location on our Web site to which you can contribute more via the "Easy Publish" menu in WebScripter.

Let's assume that a second user comes along later whose job it is to maintain a list of researchers with Semantic Web expertise, plus their email addresses and home pages. She starts WebScripter in the same way as above, selects for example PhDStudent and adds "name" as the first column in her report. At that point, WebScripter will not only add all instances of Person, but also automatically align their names into the column. Similarly, when then selecting the email address for either Person or PhDStudent and saying "Add as new column" WebScripter will fill in the email addresses for the other ontology as well. This will not happen after she adds *Has-Home-Page* as a new column (as there is no existing equivalency data) so that she has to manually select *homepage* and say "Add to column". (However, if she is willing to share her alignment data via the "Easy Publish" option future users do not have to align this column by hand either.)

5. THOUGHTS ON INCENTIVIZING PRODUCERS

As of the time of writing, one issue we encountered is that

Name	First	Last	Affiliation	Role	Email	Phone	Interests
Gio Wiederhold			organization:Stanford University				
Stefan Decker			organization:Stanford University				
Steffen Staab					sst@aifb.uni-karlsruhe.de	+49-(0)721-608 4751	
Professor James A. Hendler, PhD					jhendler@darpa.mil		
Sheila McIlraith				Research Scientist	sam@ksl.stanford.edu	650-723-7932	
Deborah L. McGuinness				Associate Director	dlm@ksl.stanford.edu	650-725-5850	
	Martin	Frank		researcher		310-448-9182	model-based user interfaces, demonstration, end-user inte semantic web, adaptive real- resource allocation
Siegfried Handschuh	Siegfried	Handschuh			handschuh@acm.org	+49 (0)721 608 7363	
Yuhui Jin	Yuhui	Jin			yhjin@db.stanford.edu	(650) 725 3321	
	Juan	Lopez		student		310-448-8276	Adaptive User Interfaces, Hu Interaction, Internet and Mul

Figure 9: Snapshot of (a fragment of) a large-size WebScripter DAML people report.

there is not that much interesting, continuously updated RDF(S), much less DAML, available on the Web today.[4] What made the original Web take off was that there was an immediate incentive for producers to use the technology because it was an easy way to publish information. We currently see no strong motivation for producers to put work into putting out RDF(S) in addition to their regular HTML pages, but there is at least a compelling intra-organizational benefit in using RDF(S) and WebScripter to generate regular HTML pages by pulling RDF from various pages within the organization.

To be more concrete, once a DAML-enabled document is published on the Web, WebScripter makes it easy to access and republish portions of it as part of a larger report – an effort savings for federated information providers who currently need to maintain the same information in multiple places. For example, professors routinely publish a list of their publications on their home page. Departments publish a list of all publications, and project pages publish a list of project-related publications from the project members. Today, someone has to manually construct these pages (presuming these federated organizations are not so tightly integrated that they maintain a shared database or other common structured information source, of course). When an author publishes a new paper or makes a correction on an existing one, he or she has to either manually update the other pages, or coordinate with the appropriate people to have all the other lists updated. WebScripter can eliminate the additional work, authors only need to mark up their personal paper publication with DAML, and the reports for the department and project-specific pages will automatically pick up the new publication (e.g. every night). WebScripter eliminates overhead not only for the organization, but also for the individual producing the information, who no longer needs to coordinate the redistribution effort. WebScripter can also enhance the flexibility and value of Web sites with large amounts of information by publishing skeleton WebScripter reports that visitors can refine to obtain customized reports. Thus, we are cautiously optimistic that WebScripter may help with the adoption of RDF(S)/DAML on the producer side as well.

[4]The notable exception are headline exchange files such as *slashdot.org/slashdot.rdf*.

6. THOUGHTS ON END-USER CONTROL OVER AUTO-ALIGNMENT

You can currently run WebScripter either in an "ignore all equivalencies" mode or in an "auto-insert all known equivalencies" mode, neither of which is ideal of course. In particular, the latter may quickly become impractical if a large number of people share alignment data, even if they are not ill-intentioned. This is either because they made a honest mistake (they aligned homepages from one ontology with email addresses from another and did not notice) or because they had a different type of equivalency in mind when they authored their report (graduate research assistants are the same as machines in the sense that they cost the project money to support, but that may then cause machines to auto-appear in a report of someone else trying to author a personnel list). We see the following potential solutions (which are not mutually exclusive).

- *Centralized Human Editors.* One possiblity is for an organization to appoint an "alignment czar". The job of such a czar would be to periodically validate the equivalency data contributed by organization members into a staging area. If approved, equivalency files are then moved to that organization's official equivalency data area. Cautious organization members can then exclusively make use of the approved equivalency data while adventurous ones are free to use staging data or external data. Obviously, any use of explicit human effort is associated with costs; however, one attraction of this model is that the "alignment czar" does not nearly need the technical sophistication of an "ontology librarian" and can possibly be a clerical worker given a specialized graphical application.

- *Social Filtering.* Another approach would be to keep track of the authors of equivalency statements as well as the users of equivalency statements (neither of which we currently do); this would enable users to say "I want to use the same equivalency data that Jim and Chris are using" (this is a nicely implicit way to limit equivalencies to e.g. the accounting context if they are co-workers in accounting, without having to more formally define the context, which is a more abstract and difficult task). This would also allow cautious users to express "I am willing to use any DAML equivalency file that at least 10 others are using" (which addresses the erroneous-alignment problem but not the context mismatch problem).

- *Fine-Grained Control in the User Interface.* Finally, it would be nice to have a compact display of the available equivalency information. This display would show a row of information about the available equivalency information and give the user a checkbox for incorporating or ignoring each. Table 2 sketches a preliminary design for deciding which *sameClassAs* statements to use. (This sketch assumes that we store much more fine-grained information in the equivalency files than we currently do.)

  The first column shows the human-given label of the class that is being declared as equivalent to the one the user added by hand. The second column indicates the level of indirection - 1 if the equivalency file directly states that the class the user just added by hand is the same as the class shown, 2 or more if the equivalence was inferred by transitive closure. The third column contains the Uniform Resource Locator for the equivalency file. The fourth column shows the name of the author of the WebScripter report that implied the equivalencies. The fifth column contains the number of additional rows inserted into the user's report if she would incorporate the equivalency. The sixth column indicates when the report that resulted in the equivalency statements was authored. The last column sums up how many other users already made use of the equivalency statement in their reports.

Class	Hops	Origin	Author	Rows	Date	Users
Person	1	stanford.e...	Smith	235	10/6/02	12
Employee	1	stanford.e...	Smith	57	10/6/02	6
Staff	1	stanford.e...	Smith	697	10/6/02	0
Member	2	www.isi.e...	Chen	15	3/4/01	17
Person	2	cmu.edu/...	Miller	973	12/7/01	4
Member	2	cmu.edu/...	Miller	107	12/7/01	9

Table 2: Sketch of a graphical user interface.

7. THOUGHTS ON OTHER OPEN QUESTIONS

Addressing a number of other issues would also help in making DAML and WebScripter use take off.

- *How do ordinary users find good original Semantic Web content?* WebScripter does not address this problem: once you found one it can point you to related content that others may have by using an equivalency-aware DAML search engine such as Teknowledge's DAML Semantic Search Service [3]. There are no Yahoo-style portals for DAML content yet to our knowledge. There are, however at least two RDF crawlers – one from BBN [2] and one from the University of Karlsruhe [5] – that could help in building such a portal.

- *What does it really mean for two classes or two attributes to be "the same"?* The current DAML equivalance statements allow users to say that x is equivalent to y. We likely need a replacement construct that allows users to express that x is equivalent to y in the sense of (or context of) z. We will try to influence the DAML language definition in that direction (but admittedly aren't quite sure ourselves how to model z). The most difficult problem we see is in the end-user interface for stating these more complex equivalencies.

8. RELATED WORK

WebScripter's approach to ontology alignment is extreme: terms from different ontologies are always assumed to mean different things by default, and all ontology mapping is done by humans (implicitly, by putting them into the same column of a report).

This is similar in spirit to Gio Wiederhold's mediation approach to ontology interoperation [18], which also assumes that terms from different ontologies never mean the same thing unless committees of integration experts say they are. WebScripter pushes that concept to the brink by replacing the experts with ordinary users that may not even be aware

of their implicit ontology alignment contributions. (Note, however, that we cannot yet proof that this collective alignment data is indeed a useful source for automatic ontology alignment on an Internet scale – we lack sufficient data from distributed WebScripter use to make that claim.)

The ONION system [15] takes a semi-automated approach to ontology interoperation: the system guesses likely matches between terms of two separately conceived ontologies, a human expert knowledgeable about the semantics of both ontologies then verifies the inferences, using a graphical user interface. ONION's guessing analyzes the schema information using relationships with semantics known to the system in advance (subclass-of, part-of, attribute-of, instance-of, value-of); in WebScripter human users rely purely on the *data instances* to decide what collates and what doesn't (because they are just not expert enough to analyze the abstractions). That being said, incorporating ONION-style alignment guessing into WebScripter would clearly be beneficial presuming the rate of correct guesses is sufficiently high.

OBSERVER [14], SIMS [9], TSIMMIS [11] and the Information Manifold [13] are all systems for querying multiple data sources of different schemata in a uniform way; however, they all rely on human experts to devise the ontological mappings between the sources to our knowledge. This is because they mediate between structured dynamic data sources (such as SQL/ODBC sources) without run-time human involvement where a higher level of precision is required to make the interoperation work. In contrast, WebScripter is targeted towards mediating between different ontologies in static RDF-based Web pages with run-time human involvement, where the need for precision in the translation is naturally lower.

9. EVALUATION AND CONCLUSIONS

WebScripter has turned out to be a valuable practical tool even for the simple single-ontology case where there is only one schema but the instance data is distributed over many Web pages. For example, the Distributed Scalable Systems Division at ISI automatically pulls together its people page from many different DAMLized Web pages: some information is maintained by individuals themselves (such as their research interests), other information is maintained by the division director (such as project assignments), and some information is maintained at the institute level (such as office assignments); this relieved the administrative assistant from manually maintaining everyone's interests [6]. WebScripter has also been used externally, for example to maintain a Semantic Web tools list [7]. You can download WebScripter from [8].

However, the most exciting application of WebScripter, as a world-wide collaborative ontology translation tool, is confined to experimental use by ourselves at this point. This is more due to a lack of widespread interesting RDF(S) content than it is due to any limitation of WebScripter itself. Nevertheless, we are excited about this new approach to global knowledge sharing, may it be achieved by a future version of WebScripter or a similar tool or tools. The key difference we see between "traditional" ontology translation and our approach is that non-experts perform all of the translation - but potentially on a global scale, leveraging each others' work.

10. ACKNOWLEDGMENTS

We gratefully acknowledge DARPA DAML program funding for WebScripter under contract number F30602-00-2-0576. The first author would also like to acknowledge AFOSR funding under grant number F49620-01-1-0341.

11. REFERENCES

[1] http://www.w3.org/TR/2000/CR-rdf-schema-20000327/.
[2] http://www.daml.org/crawler/.
[3] http://reliant.teknowledge.com/DAML.
[4] http://www.isi.edu/webscripter/daml-personnel.gen.html.
[5] http://ontobroker.semanticweb.org/rdfcrawl.
[6] http://www.isi.edu/divisions/div2/. Click on People.
[7] http://tools.semanticweb.org.
[8] http://www.isi.edu/webscripter.
[9] Y. Arens, C. Knoblock, and W.-M. Shen. Query reformulation for dynamic information integration. *Intelligent Information Systems*, 6(2-3):99–130, 1996.
[10] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, May 2001.
[11] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, V. Vassalos, and J. Widom. The TSIMMIS approach to mediation: data models and languages. *Intelligent Information Systems*, 8(2):117–32, 1997.
[12] E. Hovy. Combining and standardizing large-scale, practical ontologies for machine translation and other uses. In *Proceedings of the First International Conference on Language Resources and Evaluation (LREC)*, 1998.
[13] A. Levy, D. Srivastava, and T. Kirk. Data model and query evaluation in global information systems. *Intelligent Information Systems*, 5(2):121–43, 1995.
[14] E. Mena, A. Illarramendi, V. Kashyap, and A. Sheth. OBSERVER: an approach for query processing in global information systems based on interoperation across pre-existing ontologies. *Distributed and Parallel Databases*, 8(2):223–71, 2000.
[15] P. Mitra and G. Wiederhold. An algebra for semantic interoperability of information sources. In *2nd Annual IEEE International Symposium on Bioinformatics and Bioengineering*, pages 174–82, Bethesda, MD, USA, November 4-6 2001.
[16] P. Mitra, G. Wiederhold, and M. Kersten. A graph-oriented model for articulation of ontology interdependencies. In *Advances in Database Technology - EDBT 2000. 7th International Conference on Extending Database Technology*, Lecture Notes in Computer Science, pages 86–100, Konstanz, Germany, March 27-31 2000.
[17] N. F. Noy and M. A. Musen. PROMPT: Algorithm and tool for automated ontology merging and alignment. In *17th National Conference on AI*, 2000.
[18] G. Wiederhold. Interoperation, mediation, and ontologies. In *International Symposium on Fifth Generation Computer Systems, Workshop on Heterogeneous Cooperative Knowledge-Bases*, volume W3, pages 33–48. ICOT, Tokyo, Japan, December 1994.

Easing Participation in the Semantic Web

Stefan Haustein, Jörg Pleumann
Computer Science VIII, X
University of Dortmund,
Baroper Str. 301
D-44221 Dortmund, Germany
{stefan.haustein, joerg.pleumann}@udo.edu

ABSTRACT

Although a promising idea, the Semantic Web currently seems to have a problem duplicating the success story of its predecessor, the World Wide Web. The number of people actively participating in the Semantic Web has been very limited until now, because people can't see the benefits originating from the extra effort they invest into semantically rich web pages. Unfortunately, this advantage is barely visible at all until a critical mass of RDF-annotated pages is available on the net, thus making is difficult to recruit new participants for the Semantic Web. The article tries to break this vicious circle by showing that the use of appropriate tools may both ease participation in the semantic web and provide a number of additional advantages not directly related to the Semantic Web. The latter, in particular, may convince a larger number of people to participate, and thus bring the Semantic Web nearer its critical mass.

1. INTRODUCTION

The Semantic Web is a great idea. Yet, it did not quite take off until now. Why is this the case? Some argue that RDF [19], the language for adding the semantic information to existing web pages is the problem. These critics see RDF as being too complicated or under-specified [11, 6]. While RDF truly has its problems in some areas, we don't think that the language itself is the main obstacle that hinders people from participating in the Semantic Web. But to find out where the problem actually lies, we first need to take a step back and look at what made the original web such a tremendous success.

In our opinion, there were four important reasons for the success of the World Wide Web:

- Simplicity. HTML was easily understood and quickly written down. Even novices could design a few basic web pages with little effort, put them in a matching directory structure and start an HTTP daemon to deliver the content to clients.

Semantic Web Workshop 2002 Hawaii, USA

- Immediate feedback. After an HTML page had been designed in a text-editor, the result could be displayed in any HTML client to get an impression of the results. Thus, the user had an immediate feedback on his or her work.

- Additional benefits. Even though their original purpose was to present information to other people, HTML pages could be used as a means of discussion or documentation for people participating in a project or even for personal use. Thus, there was an additional gain users got from participating in the world wide web, which made the system even more attractive to them.

- Low critical mass. As a networked effort, the World Wide Web required a minimum (but large enough) number of participants to raise the interest of outside people, convincing them to become involved. Yet, since the World Wide Web was the first system of its kind, and there was no similar system to compete with, this critical mass was relatively low.

When we compare these points to the Semantic Web in its current form, we notice that most of them are not fulfilled:

- Simplicity is only partially given. The mixture of RDF and DAML+OIL is understood in all its details only by people that have a background in AI or related fields. Novices will only be able to use basic concepts of RDF and might thus have problems to see the real advantages of the Semantic Web.

- Immediate feedback is not given. Unfortunately, there is no specific client software for the Semantic Web that gives users an impression of their RDF fact base. One could argue that it doesn't even make sense to ask for such a software, because the clients of the Semantic Web are programs rather than human beings.

- There are no additional benefits, at least none that are ovious to "'ordinary end-users"'. While human-readable HTML pages primarily designed for other people can also be used for personal purposes, this is not true for RDF facts, which are meant to be read by programs.

- The critical mass is considerably higher. Why is this the case? This time, there already is an existing system — the original World Wide Web — , and

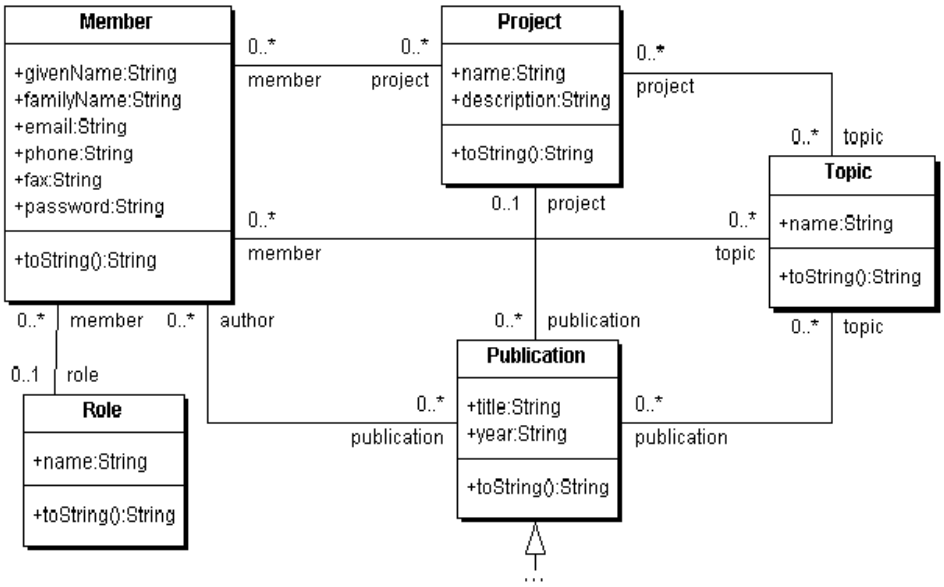

Figure 1: Simple UML Diagram for university department's web site

most people nowadays tend to use the "'brute force"' method to find a specific piece of information in it, namely Google or some other search engine. Thus, it is more difficult to convince people to take part in another system, even if it is an extension to the existing one.

As long as the first three points are true, the critical mass of users needed to make the Semantic Web "'take off"' will be hard to reach. Unfortunately, seen the other way round, the Semantic Web hardly has some kind of real benefit unless there is a large-enough number of participants that makes available RDF-specified information to others, that is, until the critical mass is reached. The current situation could be seen as some kind of vicious circle that has to be broken before the Semantic Web has a chance to succeed.

2. TOOLS TO BREAK THE CIRCLE

To break the circle, we have to get rid of as many as possible of the four problems shown in the previous section. Since we cannot lower the critical mass for mainstream acceptance of the Semantic Web (possibly by forcing people into it), we have to focus on the other three: Simplicity, immediate feedback, and additional benefits. A very promising way to achieve this seems to be the use of appropriate tools. These tools would have to ease participation in the Semantic Web, but would also have to provide some "'added value"' that makes them attractive to end-users. Obviously, when using the tools, people will also likely participate in the semantic web, even if that is not their original motivation. The following sections try to show what features these tools might offer.

2.1 Generative approach

Looking at existing tools developed for or related to the Semantic Web, for example Protégé-2000 [23] or Ontobroker [15], one notices that these are primarily designed to support ontology and fact management. Information is stored in a knowledge base providing fine grained access. The ontology is utilized to make sure that the content corresponds to the desired structure. The two systems mentioned above are able to export their fact base to an RDF representation.

While these tools aim into the right direction, they still have a problem: As long as one wants a machine-readable RDF-version of the facts as well as a human-readable HTML-version, duplicate effort is required to maintain both. Take, for example, a typical web site for a university department containing information about the department's staff, research topics, projects, and publications. A highly structured site like this is suitable for participating in the semantic web, and it can easily be modelled using a corresponding domain ontology. Yet, a change as simple as a telephone number has to be propagated to the RDF version as well as the HTML version.

Given a Semantic Web tool followed a generative approach, the situation would be easier: Assume this tool were able to incorporate regular HTML for the unstructured part of the web site, and these pages could contain placeholders for insertion of information contained in the fact base. The tool would then be able to generate the actual HTML pages automatically from the existing RDF information – or even both from a common fact base – , thus requiring the user to maintain this fact base only, at least as far as structured information is concerned. If the generation of pages takes place at run-time, we arrive at a tool that could be seen as a "'Semantic Web-enabled HTTP server"'

While the avoidance of redundancy already is a big advantage addressing simplicity, the generative approach provides other advantages that fall into the area of "'added value"':

- In contrast to editing HTML directly, a unique look and feel can easily be established for the whole site, given an appropriate template mechanism.
- In addition to HTML and RDF, other target formats

like WML and cHTML can be generated from the same fact base, lowering redundancy even further.

- In contrast to plain HTML files, ontology-based consistency checks can be performed automatically while entering data, e.g. avoiding dangling links inside the system.

2.2 Incorporation of database features

To broaden the possible target audience of our Semantic Web server, we might try to incorporate database-like features and thus position it as an alternative to a "'heavyweight"' database solution.

While relational databases with HTML-generating frontend are quite common these days (e.g. Cold Fusion [8], PHP [2], Enhydra [1] etc.), these solutions are mainly used for sites with a simple, low-dimensional structure, such as guest books or news pages (e.g. Slashdot.org). More complex domains such as university departments often still use plain HTML files for their web presentation, or make only limited use of database tables.

Here, the reason may be that a high number of tables would be required for modeling even simple ontologies, mainly because associations are not first class members of relational database systems. Revisiting the university department scenario, we need at least tables for persons, research topics, projects, and publications. Figure 1 shows a possible UML class diagram of the database's conceptual model. Since all n:n associations require separate association tables, this results in quite a lot of normalised tables (more than 10), each of which potentially contains only a very small subset of all the possible instances.

In this case, the benefit for the creator, that is, the dynamic generation of HTML or – in our case – RDF from a single set of data, does not outweigh the extra effort inherent in maintaining the tables.

Using Semantic Web tools, the picture may change significantly. For a low number of instances, the internal knowledge base provided by a Semantic Web tool may be sufficient. Associations are direcly supported, and the ontology language also allows to specify integrity constraints for them at an appropriate level. Since Semantic Web tools usually come with a generic user interface, the need to create HTML forms for editing the tables is avoided.

2.3 Incorporation of Content Management Features

Another area that a Semantic Web tool might address is content management. Content management systems, such as Hyperwave [3], Zope [5] or OpenCMS [4] provide user, version and metadata management for a set of HTML pages or binary documents in other formats such as PDF or Word. Their set of meta data, hoewever, is usually fixed and tailored to the most common needs. Here, ontology-based Semantic Web tools provide much more flexibility, and may be superior to general content management systems in domains where the meta data requirements significantly differ from the standard set provided by content management systems.

2.4 Openess to Alternative Schema Languages

In the introduction, we claimed that beneath providing no gain that becomes immediately obvious, RDF annotation is complex.

In its current form, the Semantic Web requires users to learn yet another formal description language. Users having an background in AI may be expected to be familiar with description logics and corresponding ontology modelling tools. For mainstream acceptance, though, integration of recognised standards like UML [20] may help to improve acceptance of Semantic Web tools and thus lower the entrance barrier [13]. Most students of computer science or related engineering disciplines can be assumed to be familiar with UML and modelling tools like Together or Rational Rose. These students could easily apply their modelling knowledge to the Semantic Web and thus contribute to its group of early adopters.

3. THE INFORMATION LAYER

In order to demonstrate that participation in the Semantic web actually can be simple, and that using a server based on a fine grained fact base instead of HTML- or RDF files can provide immediate gains, we have started to model our own unit's web pages accordingly. For this purpose, we used our Information Layer system, which stores data in a simple XML format that is determined by a given ontology. The information layer uses an object-oriented model for data representation. Objects consist of atomic attributes and relations to other objects. The consistency of relations in both directions is ensured automatically, avoiding inconsistencies inside the system. The concepts and relations are defined application-dependent in an external ontology definition file. All files used by the information layer are stored as XML documents.

The InfoLayer system was originally designed as an integrated information platform for software agents and human users in a conference scenario. The system was used in the COMRIS project [21] in order to make conference information available in appropriate formats to human users as well as software agents, utilizing the same underlying knowledge base. Access to the content is possible via a generic HTML interface as well as a FIPA [16] based XML interface [18]. Obviously, when information is machine readable for software agents, it is not a big leap to make this information available for the Semantic Web as well.

In the process of modelling our unit web pages, we made several improvements to our system, simplifying the use as a replacement for a "regular" web server. While there may be alternative paths appropriate for other systems, our main purpose was to show that using semantic web systems may provide direct advantages over regular web servers, even without relying on advanced features such as knowledge integration from different sources (e.g. KAON-REVERSE [17]).

3.1 XMI Import

The original version of the Information Layer system used its own proprietary XML-based ontology description language. In order to simplify the initial step of generating the application ontology, we have replaced the internal format by XMI [20], the XML based exchange format for UML diagrams. Figure 1 shows a simplified version of the UML model currently used as a basis for our unit web pages.

We have chosen UML as ontology modelling language [13] instead of RDFS [7] because it is difficult to avoid contact with UML when working in computer science or in the IT industry in general. For most computer scientists, a UML editor like Rational Rose or Together is part of their standard

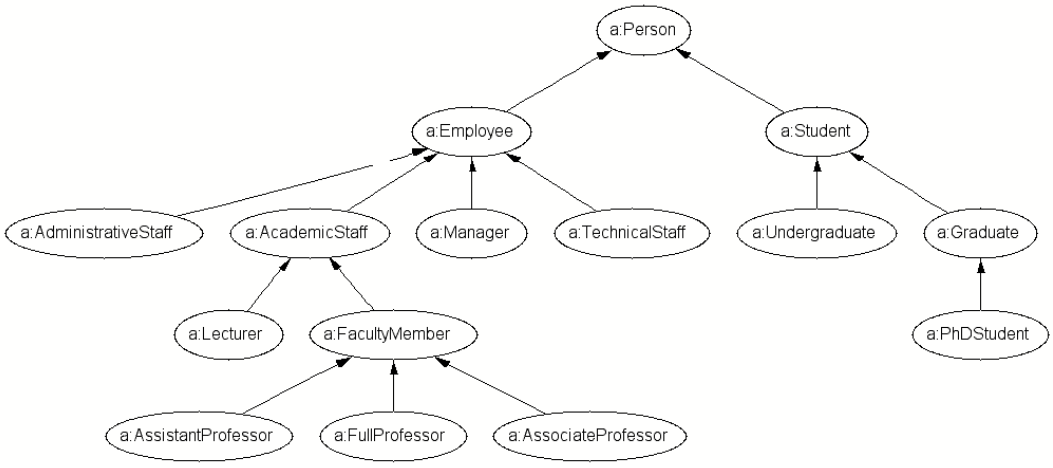

Figure 2: A Subset of the Semantic Web Research Community ontology concept Hierarchy

tool box. Thus, the extra effort of installing and getting familiar with an RDFS editor, possibly preventing people from getting in touch with the Semantic Web, is avoided.

Compared to other languages suitable for ontology modelling, UML currently still lacks clearly defined semantics. However, there are significant efforts to solve this problems [22, 10].

This aspect may be less important for systems providing their own comfortable Ontology editor.

3.2 HTML Generation

The most important capability required for being able to replace existing web servers is – of course – the generation of HTML pages.

The information layer contains a module that provides built-in web-server functionality. The server is able to generate HTML dynamically: For any object, the attributes are simply displayed, and the associations to other objects are converted to sets of hyperlinks to the related objects. Concepts are displayed as a clickable list of instances corresponding to the concept. The HTML interface can also be used to edit the content of the system using forms generated dynamically based on the ontology. In the COMRIS project, the HTML interface was used for interaction with the end user as well for as debugging and inspection purposes.

In addition to generic HTML generation, templates can be used in order to generate HTML pages conforming to a given look and feel. In the COMRIS project, we have also used the template mechanism to generate the input structure required by the text generation system TG/2 ([9]) which was used to generate natural language output for a wearable device. The template mechanism is described in some more detail in the next section.

3.3 SWRC and RDF Integration

The Semantic Web Research Community (SWRC) Ontology [24] is an ontology designed in order to describe the structure of the Semantic Web Research Community, namely the members, events, topics and projects, in a machine-readable manner. It is available in DAML+OIL and FLogic formats. Figure 2 shows a subset of the inheritance hierarchy of the SWRC ontology.

Since our "local" research unit ontology was primarily designed to fit the needs of our "regular" web presentation, it does not match the "shared" SWRC ontology exactly. However, using the template mechanism of our system, we are able to generate RDF pages corresponding to the SWRC ontology on the fly. Figure 3 shows a simplified example template that is used to generate SWRC-compliant RDF content for instances of the class "Member". In the templates, elements in a special namespace, denoted by the t prefix in the example, are replaced by content queried from the Information Layer with respect to the current instance which is determined from the page URL.

Thus, it is possible to participate in the Semantic Web without needing to extend a predefined shared ontology, which may be bloated and still not fullfill all local requirements. Instead, the domain of interest can be modelled using a lean domain specific local ontology. The SWRC person name slot illustrates the advantage of this approach: SWRC contains only one person name slot that is not split into first and last name. If the local application requires having both parts available separately, it would be necessary to duplicate the corresponding information, when building the local ontology on top of the SWRC ontology. Also, SWRC concepts like "Organization" may not be required in a local ontology covering a single organization. Information about the local organization can be stored in a single static RDF file, not bloating the local ontology.

In addition to template based RDF generation, it would be possible to generate RDF directly corresponding to the local ontology automatically [12]. However, this feature is not implemented yet.

3.4 Infrastructure Integration

For simpler integration with the existing Web server infrastructure, we changed the Information Layer implementation to become a Java Servlet instead of a stand alone program. Running the Information Layer as a Java Servlet allows smooth integration with existing Web presentations, without any hard switch. The service can simply be added where it makes most sense, and then later be extended to

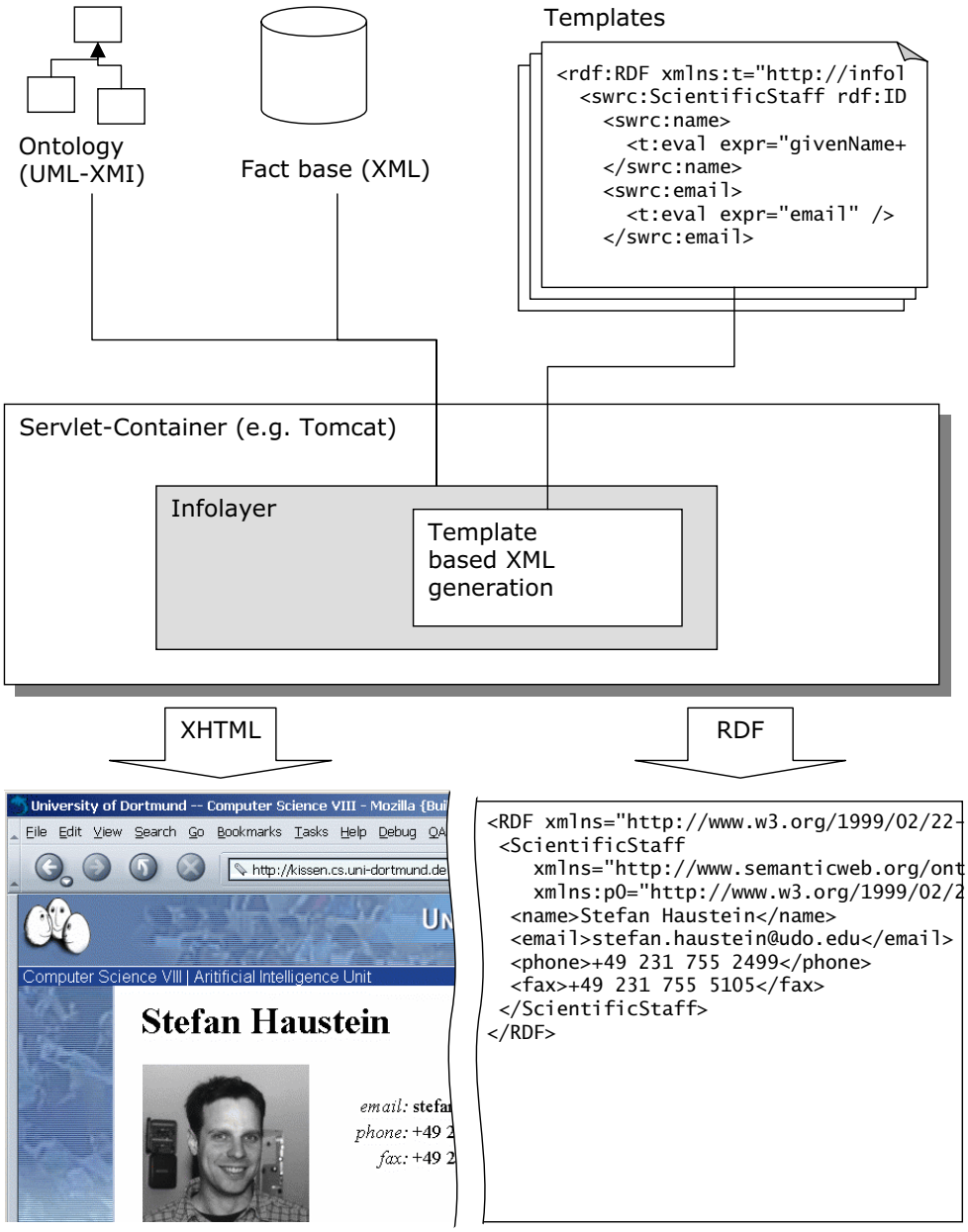

Figure 3: Dynamic HTML and RDF generation using the Information Layer template mechanism. Elements prefixed with a t: are evaluated with respect to the current instance as given in the page URL.

other areas.

3.5 File Upload

Last but not least, we have the option to upload arbitrary Files (PDF, MPG, ...) into the system. We have added this feature in order to improve suitability for general useage. While it may look a bit odd here on the first sight, it is a typical feature of content management systems. Of course, the content of the files is opaque to the system, which is controversial to the idea of providing fine grained information in RDF-format. However, the system supports the addition of relevant meta-information.

3.6 Installation

A complicated installation procedure may prevent potential users from actually using a system, even if there are obvious time or cost savings in the long run. Building on the system improvements described above, the installation of an Information Layer based system was reduced to the following steps:

1. Build a simple base ontology with the UML tool of your choice, or just use the sample ontology available from the infolayer web page as a starting point.

2. Install Apache-Tomcat or any other Web server that is capable of handling Java servlets, if not already available.

3. Install the Information Layer Servlet files in the Web services directory of the server and adopt the configuration in the `web.xml` file to your local environment.

Following these steps, a user is already able to add content using the generic Web interface and to view the content using that interface. Now the system can be further enhanced, by extending the ontology and by adding XHTML and RDF templates, customising the look and feel and the RDF generation properties of the system.

Please note that the temporal frame of the latter two steps is not fixed. For example, one could start with managing publications using the system, and then later add other concepts like projects, topics, persons or courses.

3.7 Related Systems

Obviously, other Semantic Web tools may be extended similarily. Protégé-2000 is a Knowledge Base supporting RDF format. It provides a nice Java user interface including an ontology editor, but currently lacks a plain HTML interface. Other semantic web tools such as Ontology Builder [14] or the KAON framework [17] seem to focus more on enterprise-level ontology management and information integration.

4. CONCLUSION AND OUTLOOK

The Semantic Web is a great vision. However, for a broad adoption, simple tools that allow participation without a background in AI are still rare. Protégé and similar tools seem to aim in this direction. We would like to contribute our own tool, the Information Layer. While other tools focus on easing the ontology building process, we mainly tried to address simplicity in the overall system as well as providing additional benefits that might persuade users to participate in the Semantic Web. One of these benefits is the generation of HTML as well as RDF from a common fact base to avoid redundancy, others are the incorporation of database and content management features to broaden the target audience. This way, we hope to improve availability of structured information suitable for the Semantic Web. We did not put a focus on advanced features like full DAML+OIL support, nor do not have a priority here in the future.

A web site that utilizes the Information Layer in its current form is a database for Java-enabled small devices like cell phones and personal digital assistants[1]. Here, the ontology descibes the devices, their capabilities, vendors, available protocols and known bugs. Changes to the fact base are quite frequent, but do not require the duplicated effort of updating a human and a machine-readable version, which makes the site very easy to maintain.

The Information Layer is also being used as a prototypical web presence for MuSofT, a Germany-wide project that develops multimedia teaching material for software engineering education. The site's goal is to manage and distribute the learning objects contributed by the various project partners. This installation makes use of the content management features the system provides: Learning object can be uploaded into the system from a Web browser. To allow efficient retrieval of material, LOM[2]-conforming meta-data is provided using the system's ontology capabilities.

Previous versions of the Information Layer system have been and still are are used as a basis for the MLnet teaching information server[3] and in various internal projects.

For more details about the Information Layer and its applications, please refer to `http://infolayer.org`.

5. REFERENCES

[1] The home of enhydra.org, 2001. http://www.enhydra.org.

[2] Php: Hypertext preprocessor - homepage, 2001. http://www.php.net.

[3] Hyperwave homepage, 2002. http://www.hyperwave.com.

[4] Opencms homepage, 2002. http://www.opencms.org.

[5] Zope homepage, 2002. http://www.zope.org.

[6] Dan Brickley. RDF interest group - issue tracking. Technical report, World Wide Web Consortium, 2000. http://www.w3.org/2000/03/rdf-tracking/.

[7] Dan Brickley and R. V. Guha. Ressource Description Framework (RDF) Schema specification 1.0. Technical report, World Wide Web Consortium, 2000. http://www.w3.org/TR/CR-rdf-schema-20000327.

[8] Rob Brooks-Bilson. *Programming ColdFusion.* O'Reilly, 2001.

[9] Stephan Busemann. A shallow formalism for defining personalized text. In *Workshop Professionelle Erstellung von Papier- und Online-Dokumenten at the 22nd Annual German Conference on Artificial Intelligence (KI-98)*, Bremen, September 1998.

[10] T. Clark, A. Evans, S. Kent, S. Brodsky, and S. Cook. A feasibility study in rearchitecting UML as a family of languages using a precise OO meta-modeling

[1] http://www.kobjects.org/devicedb
[2] http://ltsc.ieee.org/wg12/index.html
[3] http://kiew.cs.uni-dortmund.de:8001/

approach. Report, Precise UML Group, September 2000. http://www.cs.york.ac.uk/puml/mml/mmf.pdf.

[11] Wolfram Conen, Reinhold Klapsing, and Eckhart Koeppen. RDF model and syntax revisited: From reification to nesting, from containers to lists, from dialect to pure xml. In *International Semantic Web Working Symposium (SWWS)*, pages 195–208, Stanford University, California, USA, 2001.

[12] S. Cranefield. Networked knowledge representation and exchange using UML and RDF. *Journal of Digital information*, 1(8), 2001.

[13] S. Cranefield and M. Purvis. Uml as an ontology modelling language. In *Proceedings of the Workshop on Intelligent Information Integration, 16th International Joint Conference on Artificial Intelligence (IJCAI-99*, 1999.

[14] Aseem Das, Wei Wu, and Deborah L. McGuinness. Industrial strength ontology management. In *International Semantic Web Working Symposium (SWWS)*, pages 17–37, Stanford University, California, USA, 2001.

[15] Stefan Decker, Michael Erdmann, Dieter Fensel, and Rudi Studer. Ontobroker: Ontology based access to distributed and semi-structured information. In R. Meersman and other, editors, *Semantic Issues in Multimedia Systems, Kluwer Academic Publisher, Boston, 1999*. Kluwer Academic Publisher, Boston, 1999.

[16] Foundation For Intelligent Physical Agents (FIPA). *FIPA Agent Management Specification*, 2000. http://www.fipa.org/specs/fipa00023/XC00023F.pdf.

[17] Siegfried Handschuh, Alexander Maedche, Ljiljana Stojanovic, and Raphael Volz. KAON - the KArlsruhe ONtology and Semantic Web infrastructure. http://kaon.semanticweb.org, 2002.

[18] Stefan Haustein. Utilising an ontology based repository to connect web miners and application agents. In *Proceedings of the ECML/PKDD Workshop on Semantic Web Mining*, September 2001.

[19] Ora Lassila and Ralph R. Swick, editors. *Resource Description Framework (RDF) Model and Syntax Specification*. World Wide Web Consortium, 1999. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222.

[20] Object Management Group. OMG Unified Modeling Language Specification, version 1.3. http://www.omg.org/technology/documents/formal/unified_modeling_language.htm, 2000.

[21] Enric Plaza, Josep Llus Arcos, Pablo Noriega, and Carles Sierra. Competing agents in agent-mediated institutions. *Personal Technologies Journal*, 2(3):1–9, 1998.

[22] Precise UML Group. The Precise UML Group home page. http://www.puml.org, 2001.

[23] Stanford University. *Using Protégé-2000 to Edit RDF*, June 2000. http://protege.stanford.edu/protege-rdf/protege-rdf.html.

[24] York Sure. Swrc - semantic web research community ontology, 2001. http://ontobroker.semanticweb.org/ontos/swrc.html.

Semantics for Web-Based Mathematical Education Systems

The ActiveMath group:
Erica Melis, Jochen Büdenbender, Georgi Goguadze, Paul Libbrecht, Carsten Ullrich
melis, jochen, george, paul, cullrich @activemath.org
Universität des Saarlandes, D-66123 Saarbrücken, Germany

ABSTRACT

Web-based user-adaptive learning environments suggest a semantic knowledge representation that can be reused in different contexts. Moreover, if these educational systems employ external service systems for support or for exploratory activities, the semantic representation is a basis for the interoperability of the service systems and for machine-understandable data. ActiveMath is such a learning environment for mathematics. We show what its annotated semantic knowledge representation, extended OMDoc, is like and how it is used. We also discuss the current bottleneck of authoring. Since mathematicians are mostly familiar with authoring LaTeX rather than semantic XML, ActiveMath offers a LaTeX2OMDoc tool. As compared with the direct OMDoc authoring which is not yet visually supported this has pros and cons.

1. INTRODUCTION

Many educational systems and on-line documents have been produced in recent years. Since the encoding of the domain knowledge for a learning environment is a very expensive and time-consuming task, *reusability* of the encoded knowledge in different contexts and for different functionalities is desirable. Therefore, the representation needs to incorporate an ontology of the domain or, even better, a unique and extensible semantics of the domain concepts and their various relationships. Similarly, *inter-operability* is a prerequisite for multiple services used in education systems that can access and work with common knowledge sources.

For Semantic Web applications, mathematics is a good field to experiment with because it is largely formalized and has a clear fundamental semantics independent of presentational issues and because mathematics is a relatively well-structured field. For mathematics, an ontology needs to be enhanced by real semantics because mathematical knowledge is inherently different from its presentation (e.g., its printed version). Different presentations can mean the same thing, e.g., $\frac{1}{2}$ or 1/2. Conversely, the same presentation can mean different things in different contexts, e.g., $(\frac{ab}{p}) = (\frac{a}{p})(\frac{b}{p})$ is false in elementary algebra but true in the theory of quadratic residues.

Now, our learning environment ActiveMath [8] is a Semantic Web application for mathematics learning. Its knowledge representation is separated from its functionalities. Its knowledge representation meets the above requirements for mathematical content representations and those for educational applications that include the above mentioned reusability and inter-operability as well as the representation of pedagogical information.

ActiveMath' knowledge representation is based on OpenMath [3], a general, standardized, semantic XML-representation for mathematics. ActiveMath' functionalities require additional information to be encoded into the knowledge representation, e.g., structural information such as is-a-definition and pedagogical information such as the difficulty of an exercise.

This article shows how Semantic Web issues such as machine-understandable representation, reusability, extensibility, and migration of other representations are tackled in ActiveMath. It focuses on the knowledge representation and its current authoring. It summarizes which information is represented in the OMDoc-language which is an extension of OpenMath. It describes and substantiates the extensions we have added for the educational and other purposes of ActiveMath. It discusses how content is authored presently in a situation, where tools for semantic representations are emerging only and where the habits of authors still oppose such an encoding.

2. SEMANTIC REPRESENTATION

Although today's most common representation for knowledge of web-based systems is the syntactic markup language HTML, for a meaningful reuse in different contexts and knowledge sharing the XML representation is essential and the RDF [1] framework with data representing relations between elements can serve as a basis for building an ontology.

2.1 Semantics in Mathematical Knowledge

OMDoc has evolved historically as a standard for mathematical knowledge representation, which we decided to use for our educational system. Because of this history several features have still to be adapted to semantic Web developments, e.g. RDF. However, a big advantage of using OMDoc is its truly semantic flavour.

[1] http://www.w3.org/RDF/

Semantic Web Workshop 2002 Hawai, USA

What is the information and elements needed for mathematics? To ensure the inter-operability of mathematical systems, a keyword-annotation that might suffice for simple search functionalities is not sufficient anymore. A machine-readable input for mathematical systems such as Computer Algebra Systems (CASs) and theorem provers requires a mathematical semantics. That is, to provide a basis for multiple systems, the actual mathematical objects/formulas have to be represented. Candidates for the representation of mathematical objects and concepts are the XML-languages OpenMath[2] and content-MathML.[3] OpenMath is a (European) standard for the semantic representation of mathematical formula expressions. It semantically defines a set of mathematical symbols in the OpenMath content dictionaries and can import content dictionaries into others.

However, an extension of OpenMath is needed because (1) an ontology based on OpenMath lacks most relations except of theory-inclusion, (2) the OpenMath content dictionaries are incomplete and a simple extension mechanism is needed, (3) OpenMath has no means to structure the content of a mathematical document by dividing it into its logical units such as "definition", "theorem", and "proof".

For these reasons, OpenMath has been extended to OMDoc [6] which includes structure markup for mathematical concepts such as definitions and theorems and for other items such as examples, exercises, and elaborative texts. It also allows to define new symbols. OMDoc items may contain metadata, formal elements, textual elements, and references. References can be concept identifiers, URLs, and additional code elements. Metadata in OMDoc represent legal information compliant to Dublin Core metadata [11] and extradata element for metadata extensions. OMDoc allows to represent some mathematical dependencies: morphisms between theories, equivalence of definitions, proof-for. Implicitly it contains a dependency of symbols by the occurrence of symbols in another symbol definition.

For our application the OMDoc metadata and relations between elements are insufficient, because, on one hand, ActiveMath needs a pedagogical ontology. For its use in learning environments, we have extended OMDoc. On the other hand, relations between elements such as mathematical dependency, corollary of a theorem, similar examples, counterexample for a concept, which we introduce are also general for the field rather than due to the educational application,but are not present in OMDoc. Some of our extensions are not specific for tutorial applications, such as technical requirements[4] and bibliographical references.

2.2 Pedagogical Knowledge

The extensions described in the following are motivated by the tutorial application and generally applicable for learning systems rather than specific for a mathematics system.[5] They include, among others

- pedagogical *properties* such as difficulty. They are relevant for ActiveMath' user-adaptivity because these metadata allow to present materials that fit the current learning situation;
- pedagogically motivated *relations* among the different pieces of knowledge such as is-prerequisite. This information can be used, e.g., to present the learner prerequisites for understanding a concept, to generate links, and to generate an adaptively chosen and structured learning content.

On the one hand, metadata standards for learning resources [6] contain too many metadata which are not relevant for our purpose. On the other hand, adaptively presenting content requires some metadata which are not yet specified in LOM. This is not surprising, since IMS[7] will be soon extended by Educational Modelling Language (EML)[8] metadata. Therefore, the ActiveMath metadata extensions of the OMDoc DTD include some metadata from LOM as well as some others. For example, we extend the list of possible values of the type of attribute of LOM metadata element relation as described below. More detailed, the pedagogical metadata defined in the ActiveMath-DTD are field, abstractness, difficulty, learning-context which belong to the OMDoc in Figure 1.

```
<definition id="def_order">
   <metadata>
      <Title xml:language="en">
        Definition of the order of a group element
      </Title>
      <extradata>
        <field use="mathematics"/>
        <abstractness level="neutral"/>
        <difficulty level="easy"/>
        <learning-context use="univ_first_cycle"/>
        <relation type="for">
          <ref theory="Th1" name="order"/>
        </relation>
        <relation type="depends_on">
          <ref theory="Th1" name="group"/>
        </relation>
      </extradata></metadata>
   <CMP xml:language="en" verbosity="3">
  ...  </CMP>
</definition>
```

Figure 1: Excerpt from an OMDoc for a definition

field describes the field to which the content of the item belongs. It enables ActiveMath to present items from particular fields (such as statistics, physics, or economy), if this is required by a pedagogical strategy. For instance, if students from different groups learn statistics, e.g., technicians, mathematicians, psychology students, they obtain different (motivating) examples and exercises from the appropriate field. The meaning of abstractness and difficulty is self-explaining. They serve to adapt the document to the learner's cognitive capabilities and learning progress. Currently, these metadata can have one of three different values. learning-context specifies for which learning context the material was intended originally. The possible values of

[2] http://www.openmath.org/

[3] http://www.w3.org/Math/

[4] For client-server applications, the annotations have to include the technical requirements to display or to invoke a material. They need to be known, in particular, to ensure that multimedia material is only offered, if the computer on the learners client can handle it.

[5] For a complete definition of the OMDoc-extensions see [2]

[6] such as the Learning Object Metadata (LOM) standardized by IEEE-LTSC (http://ltsc.ieee.org/)

[7] http://www.imsglobal.org/

[8] http://eml.ou.nl/

learning-context are those defined in the IMS-metadata standards. This information is important in case material from different sources is merged for a new course. Furthermore, the OMDoc in Figure 1 has a verbosity-attribute. The information about the verbosity of the textual parts allow the generation of different kinds of document such as slides and more verbose scripts.

The following metadata defined in the ActiveMath-DTD characterize exercises more precisely

- the targeted mastery-level of the exercise. Its values can be knowledge, comprehension, application, or transfer
- the task of the learner which can be calculate, check, explore, give_example, model, or prove
- level of interactivity and average learning time
- the technical type of the exercise. The ActiveMath-DTD allows for the values provide gap, mupad, maple, omega, multiple_choice, and fill_in currently.

ActiveMath employs these metadata to user-adaptively select exercises for a document and in the suggestion mechanism according to a particular teaching strategy that targets a particular mastery-level and adaptively supports skill acquisition.

The metadata relation is used to represent several relationships between OMDoc items. The type of this relation is specified in the type-attribute which can have the following meanings:

- the previous knowledge required to understand the item. For instance, the element in Figure 1 depends-on establishes dependencies on previous concepts.
- the similarity between the item and another one, e.g., for two examples, definitions, exercises etc. that are similar as, e.g., shown in Figure 2.
- a for-relationships which can be used in order to characterize an item with an additional functionality. For instance, an item that is a proof for a theorem could as well be an example for a method application or an example could be a counterexample for a concept.
- a citation-relationship referring to an additional bib-extra element which is defined in ActiveMath-DTD to enable a full featured citation mechanism.

One can argue that our XML specifications are "heavy on attributes". There are some pros and cons for this. Pros: when using attributes one can fix the default values for them, whereas the for body of an element we can only specify the type of data that can be placed inside (e.g. child elements, PCDATA etc.). Cons: no direct standards compliance (translation needed). Also note that, when using attributes, the need to interprete the labels is not introducing any additional effort for XML data manipulation engines.

The developement of ActiveMath metadata will be continued by: separating the metadata and the content databases; IMS content packaging (as soon as EML is integrated in IMS).

```
<definition id="def_leftcosets">
   <metadata>
      <Title xml:language="en">
        Definition of left cosets
      </Title>
      <extradata>
        <relation type="for">
          <ref theory="Th1" name="leftcosets"/>
        </relation>
           ...
        <relation type="similar">
          <ref xref="def_rightcosets"/>
        </relation>
      </extradata></metadata>
   <CMP xml:language="en">...</CMP>
</definition>
```

Figure 2: The relation to a similar definition

3. SUPPORT FOR AUTHORING

Authoring ontological XML is investigated in several projects, e.g., the SemanticWeb project[9] and the Ontology Editor Protégé [9].[10]

Authoring tools for the described truly semantic (mathematical) XML are still insufficiently developed. Such tools will not only have to support the author in employing or extending a (mathematical) ontology but also in authoring or choosing pedagogical metadata, in authoring exercises with external service systems, and support her in writing (abstracted) mathematical formulas that later can be presented via style sheets. This is a serious bottleneck currently.

So, what are the realistic alternatives currently? First, use a still preliminary tool, QMath, briefly described in §3.1 that supports the authoring of mathematical formulas. QMath provides at least some support but is not sufficiently comfortable for the average author. Second, translate from a syntactically oriented macro-based encoding and heuristically add semantics as described in §3.2. Third, wait until the open-source community including our group has produced a nice visual tool (some of it exists already, e.g., our visual editor for tables of content).

3.1 QMath

QMath is a tool and a migration format for producing OMDoc documents. It was developed by Alberto Gonzales Palomo and is currently used by ActiveMath authors to write OMDocs. A QMath document is easy to produce. This is partially due to the fact that QMath supports unicode and the author is free to write her formulas directly by using unicode symbols. Look, for instance at the following example, corresponding to the LaTeX source from the Figure 4:

```
Definition:[<-df1]
:"The definition of cartesian product"
:for:[cartesian_product]
:depends_on:[def_pair]
...
$M × N Df suchthat(pair(x,y),x ∈ M & y ∈ N)$.
```

Figure 3: Excerpt from a QMath document

The author can define a context for a document. A context contains user-defined shortcuts for OMDoc elements. It also contains the references of symbols used in the document

[9] http://www.semanticweb.org/

[10] http://www.smi.stanford.edu/projects/protege/

to symbols defined in an OpenMath content dictionary. When the author uses a symbol, QMath suggests the symbol declaration procedure which assigns a meaning to the symbol by an explicit reference to a symbol in a content dictionary or via a pre-recorded context.

QMath supports metadata elements of OMDoc as well as some additional metadata of ActiveMath. This metadata support can be extended. As soon as a new element is declared in a document or in a context QMath performs a transformation to XML markup. [11]

3.2 LaTeX2OMDoc

For handling macro-based encodings, we face *two* demands: (1) the migration of existing mathematical learning document sources encoded in presentational languages, say LaTeX, and (2) the new encoding by authors/maths professors who are used to writing LaTeX and oppose authoring a truly semantic representation. In the following, we describe our efforts in both directions.

The migration of an existing large LaTeX document was the goal of a case study we conducted in 2001. The LaTeX sources of the document [4] were not intended to be migrated to a semantic representation originally. It unrestrictedly uses author-specific macros. Although the LaTeX source was already split into 'slices' and provided some dependencies, the document was designed pretty linearly rather than appropriate for a hypertext presentation and it was difficult to prepare for a reuse. For instance, it included text such as *"As we have seen in the previous example..."*.

The logical structure of the in the existing LaTeX document had to be carefully redesigned in order to obtain independently reusable items related via dependencies. For instance, many basic pieces in the text still included another one. E.g., introductions or elaborations contained a definition.

Another problem was the use of not-represented abbreviations. The text contained elements like ... *the correct notation for this should be* $\frac{\partial f}{\partial x}(a)$ *but we shall write only* $\frac{\partial f}{\partial x}$ *since it is clear that we are talking about the derivative in the point* a. A semantic representation would need to refer to the same mathematical object that may have different annotations, e.g., abbrev and a default.

The purely syntactic use of notation is one of the major problems. In a sentence like *Be carefull with our notations! In some cases* (a, b) *will mean an open interval and otherwise just an ordered pair.* (a, b) purely syntactical and its semantics is context-dependent. An automatic translation is almost impossible or at least has a highly context-dependent heuristics. The presentation-oriented representation and mis-use of LaTeX is another problem. This is obvious when the LaTeX encoding `$\{x: x\in A$ und $x$ ist rational $\}$` of the formula $\{x : x \in A$ und x ist rational$\}$ is analyzed and shows that a mathematical formula is scattered into pieces and combined again by natural language text.

Apart the context-dependent and presentation-oriented representation, presentational and semantic information is mixed especially in the mathematical formulas and therefore heuristics for correctly parsing all formulas cannot be provided. This makes a fully automated translation practically impossible.

Our experience suggests that it is impossible to translate a LaTeX source automatically that has been written without the goal of a semantic representation in mind. This not only requires to implement many document-specific heuristics, it boils down to about 50% manual translation. Even worse, often the original encoding does not allow a unique translation to semantically represented formulas and may be author's work again.

A solution can only rely on a quasi-semantic markup in LaTeX that uses macros and environments to encode information needed for semantic knowledge representation and is extensible. This has been attempted in another case study, where we instructed 'conservative' authors to write strictly defined LaTeX sources and provided a tool for an automatic conversion to OMDoc via QMath.

We defined LaTeX macros and environments for the representation of semantic information and meta-data to support the automatic conversion to OMDoc. If an element has non-empty children, it is encoded by an environment, otherwise a macro is used[12]. We specified restrictions – in particular for writing formulas in that LaTeX. The most important restrictions are

- use the symbols already defined in OpenMath content dictionaries in order to ensure reusability,
- specify the interpretation of source formulas written in LaTeX, e.g. infix or prefix notation
- use the pre-defined LaTeX environments and macros for defining OMDoc elements, i.e., structure elements and metadata.

The following is an example of using special OMDoc-oriented LaTeX environments:

```
\begin{definition}{df1}{cartesian_product}
{The definition of cartesian product}
\depends-on{def_pair} ...
$M \times N \Df \suchthat{\pair{x}{y}}
{x \in M \and y \in N }$.
\end{definition}
```

Figure 4: Example of writing restricted LaTeX

- create a separate file to define new symbols and add XSL presentational information to the defined symbols, also define own DTD extension if necessary and XSL presentation for it.

If these requirement are met, our tool automatically converts the restricted LaTeX sources via QMath to OMDoc.

To summarize: as compared with a direct authoring of OMDoc in QMath, authoring in a restricted and augmented LaTeX is more familiar to mathematicians even if not strictly simpler. Although the direct control of the layout of a document by editing the generated PDF-document is very attractive to authors, it keeps the presentation and looses the representation and thus destroys the semantic and metadata information needed for the Semantic Web application.

[11] For more information on QMath see http://www.matracas.org/

[12] For more details see http://www.activemath.org/~ilo/articles/presentation2content112001.ps.gz

4. USAGE IN ActiveMath

Instead of a conclusion, we want to summarize what ActiveMath is able to do with the knowledge representation and what the future activities will be in this direction.

For the ActiveMath system, the reuse of content in different contexts is particularly important because its user-adaptivity implies that the same content can be presented in different ways depending on the user and in the learning situation.

ActiveMath' user-adaptive functionalities such as the presentation of the content and the dynamic suggestion generation use the structure information and metadata annotating the units of the content.

ActiveMath has the following components: a session manager, course generator, the mathematical knowledge base, a presentation planner, a user model, a pedagogical module and external mathematical systems (ActiveMath integrates several service systems for calculation, proof, and exploration such as the proof planner ΩMEGA [7] and the Computer Algebra Systems MuPAD [10] and Maple). Here, the user model is a component to store, read and update data about the learner's profile. It contains history of the actions, a list of preferences of the user and a list of competence assessments. The user's actions are analyzed by evaluators that calculate updates of the user model.

The course generation in ActiveMath is realized as follows: requests of the user are sent from the browser via a web-server to the session manager. When the user has chosen her goal concepts and scenario, the session manager sends this request to the course generator. The course generator contacts the mathematical knowledge base in order to calculate which mathematical concepts are required for understanding the goal concepts. Then the information about the user's knowledge is requested from the user model and the collected IDs of `OMDoc` items annotated with the user's knowledge mastery-levels are entered as facts into the knowledge base of the expert system. Then the rules are evaluated and generate an instructional list of `XML` items to be presented. Here, the metadata such as difficulty are not only used to select appropriate exercises and examples for a learner but also for the evaluation of the user's activities.

The `XML` content is transformed to a format suitable for presentation via `XSL`. An `XSL` style sheet specifies the presentation of our `XML` documents, by describing how an instance is transformed into `HTML` or to LaTeX or Flash.

The semantic representation is a basis for merging content from different sources and presenting the merged content consistently.

The ActiveMath knowledge representation is providing two ontologies: the mathematical and the educational one. Mathematical ontology is also useful for other math applications. The mathematical concepts (elements of ontology) are the skeleton (macro level) of a document, and the educational ones provide the information for building the micro level structure.

Alternative Usages

ActiveMath' support for exploratory and interactive learning will be improved. This includes the investigation of elaborate search functions based on the the semantic and partially formal representation.

A next step is the machine-understandable description of mathematical operations. Such descriptions are useful in many situations, including the automated advise to a user for choosing an appropriate system to perform a task or for agent-based computations (see [5]).

Semantically represented repositories will be useful not just for learning environments but also for working mathematicians. For instance, `OMDoc`-structured repositories can improve the organization and searchability of mathematical knowledge. Today the digital libraries,have to face the manually controlled entry of author and classification information and often make use of keywords authored by reviewers. Today the search capabilities are limited to textual and keyword search.[13]

We understand our research as part of the larger European initiative for web-based mathematical knowledge representation and management. Its first workshop [1] covered topics ranging from publishing of large collections of electronic preprints to tools for managing mathematical documents. There is hope for a critical mass of content encoded in a semantic `XML` since the initiative will work on this as well as on tools to maintain and use the content data and metadata.

5. REFERENCES

[1] *Electronic Proceedings of the First International Workshop on Mathematical Knowledge Management*, September 2001.

[2] J. Büdenbender, G.Goguadze, P.Libbrecht, E.Melis, and C.Ullrich. Metadata for ActiveMath. Seki Report SR-02-01, Fachbereich 14 Informatik, Universität des Saarlandes, 2002.

[3] O. Caprotti and A. M. Cohen. Draft of the open math standard. Open Math Consortium,1998, `http://www.nag.co.uk/projects/OpenMath/omstd/`.

[4] B.I. Dahn and H. Wolters. *Analysis Individuell*. Springer-Verlag, 2000.

[5] M. Dewar and D. Carlisle. Mathematical software: the next generation? In [1]

[6] M. Kohlhase. `OMDoc`: Towards an internet standard for the administration, distribution and teaching of mathematical knowledge. In *AISC'2000*, 2000.

[7] E. Melis and J.H. Siekmann. Knowledge-based proof planning. *Artificial Intelligence*, 115(1):65–105, 1999.

[8] E. Melis, E. Andres, G. Goguadze, P. Libbrecht, M. Pollet, and C. Ullrich. ActiveMath: System description. In Johanna D. Moore, Carol Redfield, and W. Lewis Johnson, eds, *Artificial Intelligence in Education*, pages 580–582. IOS Press, 2001.

[9] N. F. Noy, R. W. Fergerson, and M. A. Musen. The knowledge model of Protégé-2000: Combining interoperability and flexibility. In *2th International Conference on Knowledge Engineering and Knowledge Management (EKAW'2000)*, 2000.

[10] A. Sorgatz and R. Hillebrand. MuPAD - Ein Computeralgebra System I. *Linux Magazin*,(12),1995.

[11] The Dublin Core Metadata Initiative, 1998, `http://purl.org/DC/`.

[13] The American Mathematical Society's E-Print service, e.g., allows the search through the reviews only which are traditionally encoded in the TeX language. Searching through formulas in this language is one of the most unpredictable tasks as formulas are encoded for presentation only.

Exploiting Synergy Between Ontologies and Recommender Systems

Stuart E. Middleton, Harith Alani, Nigel R. Shadbolt, David C. De Roure
Intelligence, Agents and Multimedia Group
Department of Electronics and Computer Science
University of Southampton
Southampton, SO17 1BJ, UK
{sem99r,ha,nrs,dder}@ecs.soton.ac.uk

ABSTRACT

Recommender systems learn about user preferences over time, automatically finding things of similar interest. This reduces the burden of creating explicit queries. Recommender systems do, however, suffer from cold-start problems where no initial information is available early on upon which to base recommendations.

Semantic knowledge structures, such as ontologies, can provide valuable domain knowledge and user information. However, acquiring such knowledge and keeping it up to date is not a trivial task and user interests are particularly difficult to acquire and maintain.

This paper investigates the synergy between a web-based research paper recommender system and an ontology containing information automatically extracted from departmental databases available on the web. The ontology is used to address the recommender systems cold-start problem. The recommender system addresses the ontology's interest-acquisition problem. An empirical evaluation of this approach is conducted and the performance of the integrated systems measured.

General Terms

Design, Experimentation.

Keywords

Cold-start problem, interest-acquisition problem, ontology, recommender system.

1. INTRODUCTION

The mass of content available on the World-Wide Web raises important questions over its effective use. Search engines filter web pages that match explicit queries, but most people find articulating exactly what they want difficult. The result is large lists of search results that contain a handful of useful pages, defeating the purpose of filtering in the first place.

Recommender systems [23] learn about user preferences over time and automatically find things of similar interest, thus reducing the burden of creating explicit queries. They dynamically track users as their interests change. However, such systems require an initial learning phase where behaviour information is built up to form an user profile. During this initial learning phase performance is often poor due to the lack of user information; this is known as the cold-start problem [17].

There has been increasing interest in developing and using tools for creating annotated content and making it available over the semantic web. Ontologies are one such tool, used to maintain and provide access to specific knowledge repositories. Such sources could complement the behavioral information held within recommender systems, by providing some initial knowledge about users and their domains of interest. It should thus be possible to bootstrap the initial learning phase of a recommender system with such knowledge, easing the cold-start problem.

In return for any bootstrap information the recommender system could provide details of dynamic user interests to the ontology. This would reduce the effort involved in acquiring and maintaining knowledge of people's research interests. To this end we investigate the integration of Quickstep, a web-based recommender system, an ontology for the academic domain and OntoCoPI, a community of practice identifier that can pick out similar users.

2. RECOMMENDER SYSTEMS

People may find articulating what they want hard, but they are good at recognizing it when they see it. This insight has led to the utilization of relevance feedback [24], where people rate web pages as interesting or not interesting and the system tries to find pages that match the interesting, positive examples and do not match the not interesting, negative examples. With sufficient positive and negative examples, modern machine learning techniques can classify new pages with impressive accuracy. Such systems are called content-based recommender systems.

Another way to recommend pages is based on the ratings of other people who have seen the page before. Collaborative recommender systems do this by asking people to rate explicitly pages and then recommending new pages that similar users have rated highly. The problem with collaborative filtering is that there is no direct reward for providing examples since they only help other people. This leads to initial difficulties in obtaining a sufficient number of ratings for the system to be useful.

Semantic Web Workshop 2002 Hawaii, USA

Hybrid systems, attempting to combine the advantages of content-based and collaborative recommender systems, have proved popular to-date. The feedback required for content-based recommendation is shared, allowing collaborative recommendation as well. We use the Quickstep [18] hybrid recommender system in this paper to recommend on-line research papers.

2.1 The Cold-start Problem

One difficult problem commonly faced by recommender systems is the cold-start problem [17], where recommendations are required for new items or users for whom little or no information has yet been acquired. Poor performance resulting from a cold-start can deter user uptake of a recommender system. This effect is thus self-destructive, since the recommender never achieves good performance since users never use it for long enough. We will examine two types of cold-start problem.

The *new-system cold-start* problem is where there are no initial ratings by users, and hence no profiles of users. In this situation most recommender systems have no basis on which to recommend, and hence perform very poorly.

The *new-user cold-start* problem is where the system has been running for a while and a set of user profiles and ratings exist, but no information is available about a new user. Most recommender systems perform poorly in this situation too.

Collaborative recommender systems fail to help in cold-start situations, as they cannot discover similar user behaviour because there is not enough previously logged behaviour data upon which to base any correlations. Content-based and hybrid recommender systems perform a little better since they need just a few examples of user interest in order to find similar items.

No recommender system can cope alone with a totally cold-start however, since even content-based recommenders require a small number of examples on which to base recommendations. We propose to link together a recommender system and an ontology to address this problem. The ontology can provide a variety of information on users and their publications. Publications provide important information about what interests a user has had in the past, so provide a basis upon which to create initial profiles that can address the new-system cold start problem. Personnel records allow similar users to be identified. This will address the new-user cold-start problem by providing a set of similar users on which to base a new-user profile.

3. ONTOLOGIES

An ontology is a conceptualisation of a domain into a human-understandable, but machine-readable format consisting of entities, attributes, relationships, and axioms [12]. Ontologies can provide a rich conceptualisation of the working domain of an organisation, representing the main concepts and relationships of the work activities. These relationships could represent isolated information such as an employee's home phone number, or they could represent an activity such as authoring a document, or attending a conference.

In this paper we use the term ontology to refer to the classification structure and instances within the knowledge base.

The ontology used in our work is designed to represent the academic domain, and was developed by Southampton's AKT team (Advanced Knowledge Technologies [20]). It models people, projects, papers, events and research interests. The ontology itself is implemented in Protégé 2000 [10], a graphical tool for developing knowledge-based systems. It is populated with information extracted automatically from a departmental personnel database and publication database. The ontology consists of around 80 classes, 40 slots, over 13000 instances and is focused on people, projects, and publications.

3.1 The Interest-acquisition Problem

People's areas of expertise and interests are an important type of knowledge for many applications, for example expert finders [9]. Semantic web technology can be a good source of such information, but usually requires substantial maintenance to keep the web pages up-to-date. The majority of web pages receive little maintenance, holding information that does not date quickly. Since interests and areas of expertise are dynamic in nature they are not often held within web pages. It is thus particularly difficult for an ontology to acquire such information; this is the *interest-acquisition* problem.

Many existing systems force users to perform self-assessment to gather such information, but this has numerous disadvantages [5]. Lotus have developed a system that monitors user interaction with a document to capture interests and expertise [16]. Their system does not, however, consider the online documents that users browse.

This paper investigates linking an ontology with a recommender system to help overcoming the interest acquisition problem. The recommender system will regularly provide the ontology with interest profiles for users, obtained by monitoring user web browsing and analysing feedback on recommended research papers.

4. Related Work

Collaborative recommender systems utilize user ratings to recommend items liked by similar people. PHOAKS [26] is an example of a collaborative filtering, recommending web links mentioned in newsgroups articles. Only newsgroups with at least 20 posted web links are considered by PHOAKS, avoiding the cold-start problems associated with newer newsgroups containing less messages. Group Lens [14] is an alternative example, recommending newsgroup articles. Group Lens reports two cold-start problems in their experimental analysis. Users abandoned the system before they had provided enough ratings to receive recommendations and early adopters of the system received poor recommendations until enough ratings were gathered. These systems are typical of collaborative recommenders, where a cold-start makes early recommendation poor until sufficient people have provided ratings.

Content-based recommender systems recommend items with similar content to things the user has liked before. An example of a content-based recommender is Fab [4], which recommends web pages. Fab needs a few early ratings from each user in order to create a training set. ELFI [25] is another content-based recommender, recommending funding information from a database. ELFI observes users using a database and infers both positive and negative examples of interest from this behaviour. Both these systems are typical of content-based recommender systems, requiring users to use the system for an initial period of time before the cold-start problem is overcome.

Personal web-based agents such as Letizia [15], Syskill & Webert [21] and Personal Webwatcher [19] track the users browsing and formulate user profiles. Profiles are constructed from positive and negative examples of interest, obtained from explicit feedback or heuristics analysing browsing behaviour. They then suggest which links are worth following from the current web page by recommending page links most similar to the users profile. Just like a content-based recommender system, a few examples of interest must be observed or elicited from the user before a useful profile can be constructed.

Ontologies can be used to improve content-based search, as seen in OntoSeek [13]. Users of OntoSeek navigate the ontology in order to formulate queries. Ontologies can also be used to automatically construct knowledge bases from web pages, such as in Web-KB [8]. Web-KB takes manually labelled examples of domain concepts and applies machine-learning techniques to classify new web pages. Both systems do not, however, capture dynamic information such as user interests.

Also of relevance are systems such as CiteSeer [6], which use content-based similarity matching to help search for interesting research papers within a digital library.

5. THE QUICKSTEP RECOMMENDER SYSTEM

Quickstep [18] is a hybrid recommender system, addressing the real-world problem of recommending on-line research papers to researchers. User browsing behaviour is unobtrusively monitored via a proxy server, logging each URL browsed during normal work activity. A nearest-neighbour algorithm classifies browsed URL's based on a training set of labelled example papers, storing each new paper in a central database. The database of known papers grows over time, building a shared pool of knowledge. Explicit feedback and browsed URL's form the basis of the interest profile for each user. Figure 1 shows an overview of the Quickstep system.

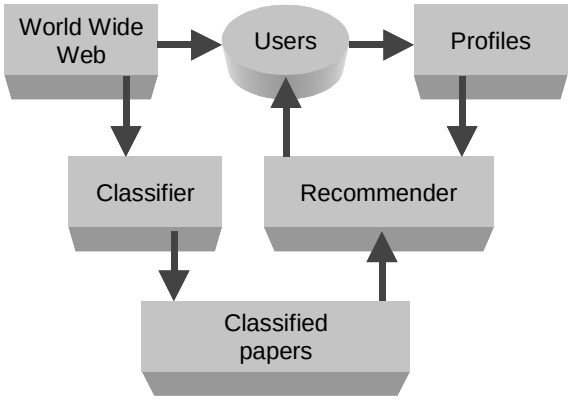

Figure 1. The Quickstep recommender system

Each day a set of recommendations is computed, based on correlations between user interest profiles and classified paper topics. Any feedback offered by users on these recommendations is recorded when the user looks at them. Users can provide new examples of topics and correct paper classifications where wrong. In this way the training set, and hence classification accuracy, improves over time.

Quickstep bases its user interest profiles on an ontology of research paper topics. This allows inferences from the ontology to assist profile generation; in our case topic inheritance is used to infer interest in super-classes of specific topics. Sharing interest profiles with the AKT ontology is not difficult since they are explicitly represented using ontological terms.

Previous trials [18] of Quickstep used hand-crafted initial profiles, based on interview data, to cope with the cold-start problem. Linking Quickstep with the AKT ontology automates this process, allowing a more realistic cold-start solution that will scale to larger numbers of users.

5.1 Paper classification algorithm

Every research paper within Quickstep's central database is represented using a term frequency vector. Terms are single words within the document, so term frequency vectors are computed by counting the number of times words appear within the paper. Each dimension within a vector represents a term. Dimensionality reduction on vectors is achieved by removing common words found in a stop-list and stemming words using the Porter [22] stemming algorithm. Quickstep uses vectors with 10-15,000 dimensions.

Once added to the database, papers are classified using an IBk [1] classifier boosted by the AdaBoostM1 [11] algorithm. The IBk classifier is a k-Nearest Neighbour type classifier that uses example documents, called a training set, added to a vector space. Figure 2 shows the basic k-Nearest Neighbour algorithm. The closeness of an unclassified vector to its neighbours within the vector space determines its classification.

$$w(d_a,d_b) = \sqrt{\sum_{j=1..T} (t_{ja} - t_{jb})^2}$$

$w(d_a,d_b)$	kNN distance between document a and b
d_a,d_b	document vectors
T	number of terms in document set
t_{ja}	weight of term j document a

Figure 2. k-Nearest Neighbour algorithm

Classifiers like k-Nearest Neighbour allow more training examples to be added to their vector space without the need to re-build the entire classifier. They also degrade well, so even when incorrect the class returned is normally in the right "neighbourhood" and so at least partially relevant. This makes k-Nearest Neighbour a robust choice of algorithm for this task.

Boosting works by repeatedly running a weak learning algorithm on various distributions of the training set, and then combining the classifiers produced by the weak learner into a single composite classifier. The "weak" learning algorithm here is the IBk classifier. Figure 3 shows the AdaBoostM1 algorithm.

Initialise all values of D to 1/N
Do for t=1..T
call weak-learn(D_t)
calculate error e_t
calculate $\beta_t = e_t/(1-e_t)$
calculate D_{t+1}

$$\text{classifier} = \underset{c \in C}{\text{argmax}} \sum_{t = \text{all iterations with result class c}} \log \frac{1}{\beta_t}$$

D_t	class weight distribution on iteration t
N	number of classes
T	number of iterations
weak-learn(D_t)	weak learner with distribution D_t
e_t	weak_learn error on iteration t
β_t	error adjustment value on iteration t
classifier	final boosted classifier
C	all classes

Figure 3. AdaBoostM1 boosting algorithm

AdaBoostM1 has been shown to improve [11] the performance of weak learner algorithms, particularly for stronger learning algorithms like k-Nearest Neighbour. It is thus a sensible choice to boost our IBk classifier.

5.2 User profiling algorithm

The profiling algorithm performs correlation between paper topic classifications and user browsing logs. Whenever a research paper is browsed that has been classified as belonging to a topic, it accumulates an interest score for that topic. Explicit feedback on recommendations also accumulates interest value for topics. The current interest of a topic is computed using the inverse time weighting algorithm shown in Figure 4.

$$\text{Topic interest} = \sum_{1..\text{no of instances}}^{n} \text{Interest value(n)} / \text{days old(n)}$$

Interest values	Paper browsed = 1
	Recommendation followed = 2
	Topic rated interesting = 10
	Topic rated not interesting = -10

Figure 4. Profiling algorithm

An is-a hierarchy of research paper topics is held so that super-class relationships can be used to infer broader topic interest. When a specific topic is browsed, fractional interest is inferred for each super-class of that topic, using a $1/2^{level}$ weighting where 'level' refers to how many classes up the is-a tree the super-class is from the original topic. Figure 5 shows a section from the research paper topic ontology.

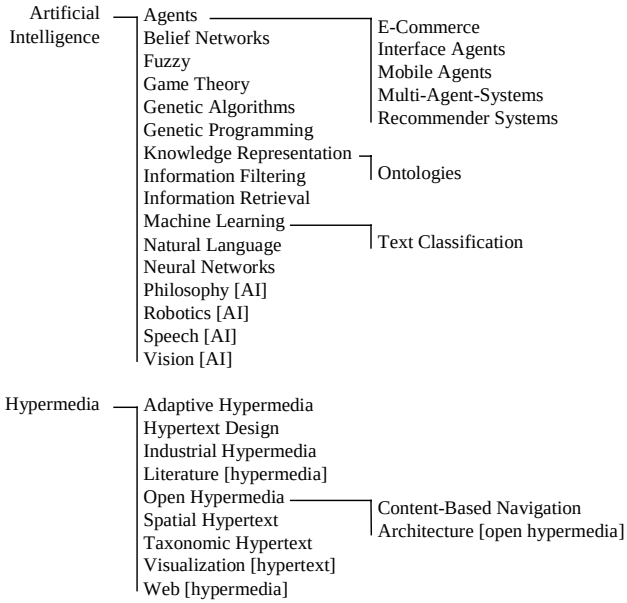

Figure 5. Section of the research paper topic ontology

5.3 Recommendation algorithm

Recommendations are formulated from a correlation between the users current topics of interest and papers classified as belonging to those topics. A paper is only recommended if it does not appear in the users browsed URL log, ensuring that recommendations have not been seen before. For each user, the top three interesting topics are selected with 10 recommendations made in total. Papers are ranked in order of the recommendation confidence before being presented to the user. Figure 6 shows the recommendation algorithm.

Recommendation confidence = classification confidence * topic interest value

Figure 6. Recommendation algorithm

6. ONTOCOPI

The Ontology-based Communities of Practice Identifier (OntoCoPI) [2] is an experimental system that uses the AKT ontology to help identifying communities of practice (CoP). The community of practice of a person is taken here to be the closest group of people, based on specific features they have in common with that given person. A community of practice is thus an informal group of people who share some common interest in a particular practice [7] [27]. Workplace communities of practice improve organisational performance by maintaining implicit knowledge, helping the spread of new ideas and solutions, acting as a focus for innovation and driving organisational strategy.

Identifying communities of practice is an essential first step to understand the knowledge resources of an organization [28]. Organisations can bring the right people together to help the identified communities of practice to flourish and expand, for example by providing them with appropriate infrastructure and give them support and recognition. However, community of practice identification is currently a resource-heavy process

largely based on interviews, mainly because of the informal nature of such community structures that are normally hidden within and across organisations.

OntoCoPI is a tool that uses ontology-based network analysis to support the task of community of practice identification. A breadth-first spreading activation algorithm is applied by OntoCoPI to crawl the ontology network of instances and relationships to extract patterns of certain relations between entities relating to a community of practice. The crawl can be limited to a given set of ontology relationships. These relationships can be traced to find specific information, such as who attended the same events, who co-authored papers and who are members of the same project or organisation. Communities of practice are based on informal sets of relationships while ontologies are normally made up of formal relationships. The hypothesis underlying OntoCoPI is that some informal relationships can be inferred from the presence of formal ones. For instance, if A and B have no formal relationships, but they have both authored papers with C, then that could indicate a shared interest.

One of the advantages of using an ontology to identify communities of practice, rather than other traditional information networks [3] is that relationships can be selected according to their semantics, and can have different weights to reflect relative importance. For example the relations of document authorship and project membership can be selected if it is required to identify communities of practice based on publications and project work. OntoCoPI allows manual selection of relationships or automatic selection based on the frequency of relationship use within the knowledge base. Selecting the right relationships and weights is an experimental process that is dependent on the ontology structure, the type and amount of information in the ontology, and the type of community of practice required.

When working with a new community of practice some experiments will be needed to see which relationships are relevant to the desired community of practice, and how to set relative weights. In the experiments described in this paper, certain relationships were selected manually and weighted based on our preferences. Further trials are needed to determine the most effective selection.

7. INTEGRATION OF THE TWO TECHNOLOGIES

We have investigated the integration of the ontology, OntoCoPI and Quickstep recommender system to provide a solution to both the cold-start problem and interest acquisition problem. Figure 7 shows our experimental systems after integration.

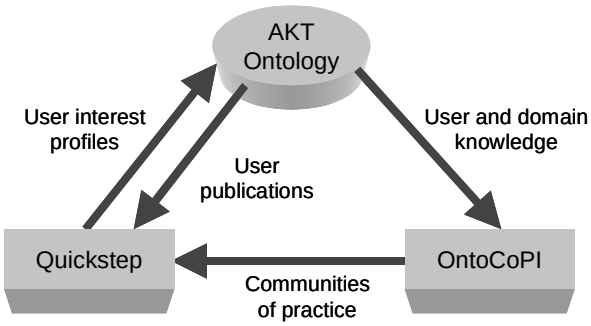

Figure 7. Ontology and recommender system integration

Upon start-up, the ontology provides the recommender system with an initial set of publications for each of its registered users. Each user's known publications are then correlated with the recommender systems classified paper database, and a set of historical interests compiled for that user. These historical interests form the basis of an initial profile, overcoming the new-system cold-start problem. Figure 8 details the initial profile algorithm. As per the Quickstep profiling algorithm, fractional interest in a topic super-classes is inferred when a specific topic is added.

$$\text{topic interest}(t) = \sum_{\substack{n \\ 1..\ \text{publications} \\ \text{belonging to class } t}} 1 / \text{publication age}(n)$$

new-system initial profile = (t, topic interest(t))*

t = <research paper topic>

Figure 8. New-system initial profile algorithm

When the recommender system is up and running and a new user is added, the ontology provides the historical publication list of the new user and the OntoCoPI system provides a ranked list of similar users. The initial profile of the new user is formed from a correlation between historical publications and any similar user profiles. This algorithm is detailed in figure 9, and addresses the new-user cold-start problem.

$$\text{topic interest}(t) = \frac{\gamma}{N_{similar}} \sum_{1..N_{similar}}^{u} \text{profile interest}(u,t) + \sum_{1..N_{pubs\ t}}^{n} 1 / \text{publication age}(n)$$

profile interest(u,t) = interest of user u in topic t * CoP confidence
new-user initial profile = (t, topic interest(t))*

t = research paper topic
u = user
γ = weighting constant >= 0
$N_{similar}$ = number of similar users
$N_{pubs\ t}$ = number of publications belonging to class t
CoP confidence = Communities of practice confidence

Figure 9. New-user initial profile algorithm

The task of populating and maintaining the ontology of user research interests is left to the recommender system. The recommender system compiles user profiles on a daily basis, and these profiles are asserted into the ontology when ready. Figure 10 details the structure of these profiles. In this way up-to-date interests are maintained, providing a solution to the interest acquisition problem. The interest data is used alongside the more static information within the ontology to improve the accuracy of the OntoCoPI system.

user profile = (topic, interest)*
topic = research topic
interest = interest value

Figure 10. Daily profiles sent to the AKT ontology

7.1 Example of system operation

When the Quickstep recommender system is first initialised, it retrieves a list of people and their publication URLs from the ontology. Quickstep analyses these publications and classifies them according to the research topic hierarchy in the ontology. Paper topics are associated with their date of publication, and the 'new-system initial profile' algorithm used to compute a set of initial profiles for each user.

Tables 1 and 2 shows an example of this for the user Nigel Shadbolt. His publications are analysed and a set of topics and dates formulated. The 'new-system initial profile' algorithm then computes the interest values for each topic. For example, 'Knowledge Acquisition' has one publication two year old (round up) so its value is 1.0 / 2 = 0.5.

Table1. Publication list for Shadbolt

Publication	Date	Topic
Capturing Knowledge of User Preferences: ontologies on recommender systems	2001	Recommender systems
Knowledge Technologies	2001	Knowledge Technology
The Use of Ontologies for Knowledge Acquisition	2001	Ontology
Certifying KBSs: Using CommonKADS to Provide Supporting Evidence for Fitness for Purpose of KBSs	2000	Knowledge Management
Extracting Focused Knowledge from the Semantic Web	2000	Knowledge Acquisition
Knowledge Engineering and Management	2000	Knowledge Management
…		

Table2. Example of new-system profile for Shadbolt

Topic	Interest
Knowledge Technology\Knowledge Management	1.5
Knowledge Technology\Ontology	1.0
AI\Agents\Recommender Systems	1.0
Knowledge Technology\Knowledge Acquisition	0.5
…	

At a later stage, after Quickstep has been running for a while, a new user registers with email address sem99r@ecs.soton.ac.uk. OntoCoPI identifies this email account as that of Stuart Middleton, a PhD candidate within the department, and returns the ranked and normalised communities of practise list displayed in table 3. This communities of practise list is identified using relations on conference attendance, supervision, authorship, research interest, and project membership, using the weights 0.4, 0.7, 0.3, 0.8, and 0.5 respectively. De Roure was found to be the closest person as he is Middleton's supervisor, and has one joint publication co-authored with Middleton and Shadbolt. The people with 0.82 values are other supervisees of De Roure. Alani attended the same conference that Middleton went to in 2001.

Table 3. OntoCoPI results for Middleton

Person	Relevance value	Person	Relevance value
DeRoure	1.0	Alani	0.47
Revill	0.82	Shadbolt	0.46
Beales	0.82		

The communities of practise list is then sent to Quickstep, which searches for matching user profiles. These profiles will be more accurate and up to date than those initially created profiles based on publications. Quickstep manages to find the profiles in table 4 in its logs.

Table 4. Profiles of similar people to Middleton

Person	Topic	Interest
DeRoure	AI\Distributed Systems	1.2
	AI\Agents\Recommender Systems …	0.73
Revill	AI\Agents\Mobile Agents	1.0
	AI\Agents\Recommender Systems …	0.4
Beals	Knowledge Technology\Knowledge Devices	0.9
	AI\Agents\Mobile Agents …	0.87
Alani	Knowledge Technology\Ontology	1.8
	Knowledge Technology\Knowledge Management\ CoP …	0.7
Shadbolt	Knowledge Technology\Knowledge Management	1.5
	AI\Agents\Recommender Systems …	1.0

These profiles are merged to create a profile for the new user, Middleton, using the 'new-user initial profile' algorithm with a γ value of 2.5. For example, Middleton has a publication on 'Recommender Systems' that is 1 year old and DeRoure, Revill and Shadbolt have interest in 'Recommender Systems' – this topics value is therefore 1/1 + 2.5/5 * (1.0*0.73+0.82*0.4+0.46*1.0) = 1.76. Table 5 shows the resulting profile.

Table 5. New-user profile for Middleton

Topic	Interest
AI\Agents\Recommender Systems	1.76
AI\Agents\Mobile Agents	0.77
AI\Distributed Systems	0.6
Knowledge Technology\Ontology	0.42
Knowledge Technology\Knowledge Devices	0.37
Knowledge Technology\Knowledge Management	0.35
Knowledge Technology\Knowledge Management\ CoP	0.16
…	

Every day Quickstep's profiles are updated and automatically fed back to the ontology, where they are used to populate the research interest relationships of the relevant people.

8. EMPIRICAL EVALUATION

In order to evaluate the effect both the new-system and new-user initial profiling algorithms have on our integrated system, we conducted an experiment based around the browsing behaviour logs obtained from the Quickstep [18] user trials. The algorithms previously described are used, as per the example in the previous section, and the average performance for all users calculated.

8.1 Experimental approach

Users were selected from the Quickstep trials whom had entries within the departmental publication database. We selected nine users in total, with each user typically having one or two publications.

The URL browsing logs of these users, extracted from 3 months of browsing behaviour recorded during the Quickstep trials, were then broken up into weekly log entries. Seven weeks of browsing behaviour were taken from the start of the Quickstep trials, and an empty log created to simulate the very start of the trial.

Eight iterations of the integrated system were thus run, the first simulating the start of the trial and others simulating the following weeks 1 to 7. Interest profiles were recorded after each iteration. Two complete runs were made, one with the 'new-system initial profiling' algorithm and one without. The control run without the 'new-system initial profiling' algorithm started with blank profiles for each of its users.

An additional iteration was run to evaluate the effectiveness of the 'new-user initial profile' algorithm. We took the communities of practice for each user, based on data from week 7, and used the 'new-user initial profile' algorithm to compute initial profiles for each user as if they were being entered onto the system at the end of the trial. These profiles were recorded. Because we are using an early prototype version of OntoCoPI, communities of practice confidence values were not available; we thus used confidence values of 1 throughout this experiment.

In order to evaluate our algorithms effect on the cold-start problem, we compared all recorded profiles to the benchmark week 7 profile. This allows us to measure how quickly profiles converge to the stable state existing after a reasonable amount of

behaviour data has been accumulated. The quicker the profiles move to this state the quicker they will have overcome the cold-start.

Week 7 was chosen as the cut-off point of our analysis since after about 7 weeks of use the behaviour data gathered by Quickstep will dominate the user profiles. The effects of bootstrapping beyond this point would not be significant. If we were to run the system beyond week 7 we would simply see the profiles continually adjusting to the behaviour logged each week.

8.2 Experimental results

Two measurements were preformed when comparing profiles to the benchmark week 7 profile. The first, profile precision, measures how many topics were mentioned in both the current profile and benchmark profile. Profile precision is an indication of how quickly the profile is converging to the final state, and thus how quickly the effects of the cold-start are overcome. The second, profile error rate, measures how many topics appeared in the current profile that did not appear within the benchmark profile. Profile error rate is an indication of the errors introduced by the two bootstrapping algorithms. Figure 11 describes these metrics.

It should be noted that we are not measuring the absolute precision and error rate of the profiles – only the relative precision and error rate compared to the week 7 steady state profiles. Measuring absolute profile accuracy is a very subjective matter, and we do not attempt it here; we are only interested in how quickly profiles reach their steady states. A more complete evaluation of Quickstep's overall profiling and recommendation performance can be found in [18].

$$\text{profile precision} = \frac{1}{N_{users}} \sum_{1..N_{users}}^{user} \frac{N_{correct}}{N_{correct} + N_{missing}}$$

$$\text{profile error rate} = \frac{1}{N_{users}} \sum_{1..N_{users}}^{user} \frac{N_{incorrect}}{N_{correct} + N_{incorrect} + N_{missing}}$$

$N_{correct}$	Number of user topics that appear in current profile and benchmark profile
$N_{missing}$	Number of user topics that appear in benchmark profile but not in current profile
$N_{incorrect}$	Number of user topics that appear in current profile but not in benchmark profile
N_{users}	Total number of users

Figure 11. Evaluation metrics

The results of our experimental runs are detailed in figures 12 and 13. The new-user results consist of a single iteration, so appear on the graphs as a single point.

At the start, week 0, no browsing behaviour log data is available to the system so the profiles without bootstrapping are empty. The new-system algorithm, however, can bootstrap the initial user profiles and achieves a reasonable precision of 0.35 and a low error rate of 0.06. We found that the new-system profiles accurately captured interests users had a year or so ago, but tended to miss current interests. This is because publications are generally not available for up-to-date interests.

As we would expect, once the weekly behaviour logs become available to the system the profiles adjust accordingly, moving away from the initial bootstrapping. On week 7 the profiles converge to the benchmark profile.

The new-user algorithm result show a more dramatic increase in precision to 0.84, but comes at the price of a significant error rate of 0.55. The profiles produced by the new-user algorithm tended to be very inclusive, taking the set of similar user interests and producing a union of these interests. While this captures many of the new users real interests, it also included a large number of interests not relevant to the new user but which were interesting to the people similar to the new user.

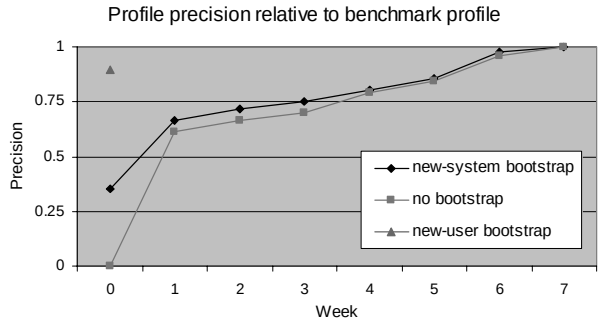

Figure 12. Profile precision

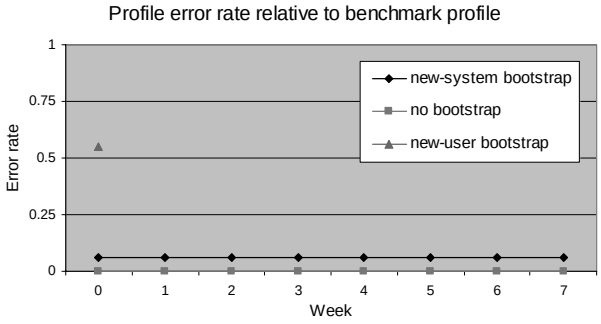

Figure 13. Profile error rate

Since error rate is measured relative to the final benchmark profile of week 7, all the topics seen in the behaviour logs will be present within the benchmark profile. Incorrect topics must thus come from another source – in this case bootstrapping on week 0. This causes error rates to be constant over the 7 weeks, since the incorrect topics introduced on week 0 remain for all subsequent weeks.

9. DISCUSSION

Cold-starts in recommender systems and interest acquisition in ontologies are serious problems. If initial recommendations are inaccurate, user confidence in the recommender system may drop with the result that not enough usage data is gathered to overcome the cold-start. In regards to ontologies, up-to-date interests are not

generally available from periodically updated information sources such as web pages, personal records or publication databases.

Our integration of the Quickstep recommender system, AKT ontology and OntoCoPI system has demonstrated one approach to reduce both the cold-start and interest-acquisition problems. Our practical work suggests that using an ontology to bootstrap user profiles can significantly reduce the impact of the recommender system cold-start problem. It is particularly useful for the new-system cold-start problem, where the alternative is to start with no information at all. Regularly feeding the recommender systems interest profiles back to the ontology also clearly assists in the acquisition of up-to-date interests. A number of issues have, however, arisen from our integration.

The new-system algorithm produced profiles with a low error rate and a reasonable precision of 0.35. This reflects that previous publications are a good indication of users current interests, and so can produce a good starting point for a bootstrap profile. Where the new-system algorithm fails is for more recent interests, which make up the remaining 65% of the topics in the final benchmark profile. To discover these more recent interests, it is possible that the new-system algorithm could be extended to take some of the other information available within the ontology into account, such as the projects a user is working on. To what degree these relationships will help is difficult to predict however, since the ontology itself has great difficulty in acquiring knowledge of recent interests.

For the purposes of our experiment, we selected those users who had some entries within the universities on-line publication database. There were some users who had not entered their publications into this database or who have yet to publish their work. For these users there is little information within the ontology, making any new-system initial profiles of little use. In a larger scale system, more sources of information would be needed from the ontology to build the new-system profiles. This would allow some redundancy, and hence improve robustness in the realistic situation where information is sparsely available.

The community of practice for a user was found not to be always relevant based on our selection of relationships and weights. For example, Dave de Roure supervises Stuart Middleton, but Dave supervises a lot of other students interested in mobile agents. These topics are not relevant to Stuart, which raises the question of how relevant the supervision relationship is to our requirements, and how best to weight such a relationship. Further experiments are needed to identify the most relevant settings for community of practice identification. The accuracy of our communities of practice are also linked to the accuracy of the research interest information as identified by the recommender system.

The new-user algorithm achieved good precision of 0.84 at the expense of a significant 0.55 error rate. This was because both the communities of practice generated for users were not always precise, and because of the new-user algorithm included all interests from the similar users. An improvement would be to only use those interests held by the majority of the people within a community of practice. This would exclude some of the less common interests that would otherwise be included into the new-user profile.

The new-user initial profile algorithm defines the constant γ, which determines the proportional significance of previous publications and similar users. Factors such as the availability of relationship data within the ontology and quality of the publication database will affect the choice of value for γ. We used a value of 2.5, but empirical evaluation would be needed to determine the best value.

There is an issue as to how best to calculate the "semantic distance" between topics within the is-a hierarchy. We make the simplifying assumption that all is-a links have equal relevance, but the exact relevance will depend on each topic in question. If individual weightings were allowed for each topic, a method for determination of these weights would have to be considered. Alternatively the is-a hierarchy could be carefully constructed to ensure equal semantic distance.

A positive feedback loop exists between the recommender system and ontology, making data incest a potential problem. For new users there are no initial interest entries within the ontology, so new user profiles are not incestuous. If the recommender system were to use the communities of practice for more than just initial profiles, however, a self-confirming loop would exist and interest calculations would be incestuous.

Finally, a question still remains as to just how good an initial profile must be to fully overcome the effects of the cold-start problem. If initial recommendations are poor users will not use the recommender system and hence it will never get a chance to improve. We have shown that improvements can be made to initial profiles, but further empirical evaluation would be needed to evaluate exactly how much improvement is needed before the system is "good enough" for users to give it a chance.

10. FUTURE WORK

The next step for the integrated system is to continue to improve the set of relationships and weights used to calculate communities of practice, and find a more selective 'new-user initial profile' algorithm. With more precise communities of practice the new-user bootstrapping error rate should fall substantially. We could then conduct a set of further user trials. This would allow the assessment of user up-take and use of the integrated system, and reveal how improving initial profiles affect overall system usage patterns.

The Quickstep recommender system is currently being extended to explore further the idea of using ontologies to represent user profiles. A large-scale trial is under way over a full academic year to evaluate the new system, which is called the Foxtrot recommender system.

11. ACKNOWLEDGEMENTS

This work is funded by EPSRC studentship award number 99308831 and the Interdisciplinary Research Collaboration In Advanced Knowledge Technologies (AKT) project GR/N15764/01.

12. REFERENCES

[1] Aha, D. Kibler, D. Albert, M. Instance-based learning algorithms, Machine Learning, 6:37-66, 1991

[2] Alani, H., O'Hara, K., and Shadbolt, N. ONTOCOPI: Methods and Tools for Identifying Communities of Practice,

Intelligent Information Processing Conference, IFIP World Computer Congress (WCC), Montreal, Canada, 2002.

[3] Albert, R. and Barabasi, AL. Statistical Mechanics of Complex Networks. Review of Modern Physics, 74, 47, 2002.

[4] Balabanovi, M., and Shoham, Y. Fab: content-based, collaborative recommendation, Communications of the ACM Volume 40, No. 3 (Mar. 1997)

[5] Becerra-Fernandez, I. Facilitating the Online Seach of Experts at NASA using Expert Seeker People-Finder. Proceedings of the 3rd International Conference on Practical Aspects of Knowledge Management (PAKM), Basel, Switzerland, 2000.

[6] Bollacker, K.D., Lawrence, S., and Giles, C.L. CiteSeer: An Autonomous Web Agent for Automatic Retrieval and Identification of Interesting Publications, Proceedings of the Second International Conference on Autonomous Agents, Minneapolis MN, USA, 1998

[7] Brown and Duguid 2000. The social life of information Harvard Buisness School Press

[8] Craven, M. DiPasquo, D. Freitag, D. McCallum, A. Mitchell, T. Nigam K. and Slattery, S. Learning to Extract Symbolic Knowledge from the World Wide Web, Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98), 1998

[9] Dunlop, M. D. Development and evaluation of clustering techniques for finding people. Proceedings of the 3rd International Conference on Practical Aspects of Knowledge Management (PAKM), Basel, Switzerland, 2000.

[10] Eriksson, H., Fergeson, R., Shahr, Y., and Musen, M. (1999). Automatic generation of ontology editors. 12th Workshop on Knowledge Acquisition, Modelling, and Management (KAW'99), Ban, Alberta, Canada.

[11] Freund, Y. Schapire, R.E. Experiments with a New Boosting Algorithm, Proceedings of the Thirteenth International Conference on Machine Learning, 1996

[12] Guarino, N., and Giaretta, P. (1995). Ontologies and Knowledge bases: towards a terminological clarification. Towards Very Large Knowledge Bases: Knowledge Building and Knwoledge Sharing. N. Mars, IOS Press: 25-32.

[13] Guarino, N., Masolo, C. and Vetere, G. OntoSeek: Content-Based Access to the Web, IEEE Intelligent Systems, Vol. 14, No. 3, May/June 1999

[14] Konstan, J.A., Miller, B.N., Maltz, D., Herlocker, J.L., Gordon, L.R., and Riedl, J. GroupLens: applying collaborative filtering to Usenet news, Communications of the ACM Volume 40, No. 3 (Mar. 1997)

[15] Lieberman, H. Letizia: An Agent That Assists Web Browsing, Proceedings of the 1995 International Joint Conference on Artificial Intelligence, Montreal, Canada, August 1995

[16] Lotus. Locating Organisational Expertise with the Lotus Discovery Server, White Paper, 2001.

[17] Maltz, D. Ehrlich, E. Pointing the way: Active collaborative filtering, CHI'95 Human Factors in Computing Systems, 1995

[18] Middleton, S.E., De Roure, D. C., and Shadbolt, N.R. Capturing Knowledge of User Preferences: ontologies on recommender systems, In Proceedings of the First International Conference on Knowledge Capture (K-CAP 2001), Oct 2001, Victoria, B.C. Canada.

[19] Mladenic, D. Personal WebWatcher: Implementation and Design, Technical Report IJS-DP-7472, Department of Intelligent Systems, J.Stefan Institute, Slovenia, 1996

[20] O'Hara, K., Shadbolt, N., and Buckingham Shum, S. The AKT Manifesto, 2001. www.aktors.org/publications/Manifesto.doc

[21] Pazzani, M. Muramatsu J. Billsus, D. Syskill & Webert: Identifying interesting web sites, Proceedings of the National Conference on Artificial Intelligence, Portland, Oregon, 1996

[22] Porter, M. An algorithm for suffix stripping, Program 14 (3), July 1980, pp. 130-137

[23] Resnick, P. Varian, H. R. Recommender systems, Communications of the ACM, 40(3), 1997, 56-58

[24] Rocchio, J.J. Relevance feedback in information retrieval, in the SMART Retrieval System -- Experiments in Automatic Document Processing, Englewood Cliffs, NJ, 1971. Prentice Hall, Inc. pp. 313-323

[25] Schwab, I., Pohl, W., and Koychev, I. Learning to Recommend from Positive Evidence, Proceedings of Intelligent User Interfaces 2000, ACM Press, pp 241-247.

[26] Terveen, L., Hill, W., Amento, B., McDonald, D., and Creter, J. PHOAKS: a system for sharing recommendations, Communications of the ACM Volume 40, No. 3 (Mar. 1997)

[27] Wenger, E. C., and Snyder, W. M. Communities of Practice: The Organizational Frontier. Harvard Business Review. January-February: 139-145. 2000

[28] Wenger, E. Communities of practice: the key to knowledge strategy. Knowledge Directions: The Journal of the Institute for Knowledge Management, 1, 48-93, 1999.

EDUTELLA: Searching and Annotating Resources within an RDF-based P2P Network

Wolfgang Nejdl, Boris Wolf
L3S and Knowledge Based Systems
University of Hannover
30167 Hannover, Germany
{nejdl,wolf}@kbs.uni-hannover.de

Steffen Staab, Julien Tane
L3S and Institute AIFB
76128 Karlsruhe, Germany
{sst,jta}@aifb.uni-karlsruhe.de

ABSTRACT

P2P applications for searching and exchanging information over the Web have become increasingly popular. This has lead to a number of (usually thematically) focused communities, which allow efficient searching within such communities, and which use specific metadata sets to specify the resources stored within the P2P network. By concentrating on domain and application specific formats for metadata and query languages, however, current P2P networks appear to be fragmenting into non-interoperable niche markets. This contribution describes the open source project Edutella which builds upon metadata standards defined for the WWW and aims to provide an RDF-based metadata infrastructure for P2P applications, building on the recently announced JXTA Framework. We describe one basic service (query) and an Edutella application (annotation) within this network, both being built on a common query language exchange format, and specify the main architecture and APIs of the Edutella P2P network.

1. BACKGROUND

The advantage of the WWW is that it constitutes a predominantly decentral paradigm storing information resources in hypertext like structures. Searching in the WWW, however, typically follows a client-server model, viz. browser vs. search engine [16], inheriting the corresponding benefits and pitfalls. To name some problems, search engines cover only a decreasing percentage of the information available on the Web and their content is often not up to date because of the time required for crawling of the Web.

In contrast, information resources in P2P networks are stored on numerous peers waiting to be queried for these resources. The querying of peer-to-peer networks allows the comprehensive retrieval of up-to-date resources stored at relevant sites. But in order to achieve this, it requires a query mechanism using some description of the resources managed by these peers.

While in the server/client-based environment of the World Wide Web metadata are useful and important, for *Peer-to-Peer (P2P)* environments that come without underlying hypertext structures metadata are absolutely crucial. Such metadata are easy to provide for specialized cases, but non-trivial for general applications. The core concern of our research therefore is to develop a general infrastructure for combining metadata with P2P networks.

In the context of educational resources for example, which we are currently focusing on, P2P-based approaches are more flexible than centralized approaches like Client-Server computing, with several advantages for the participating institutions. As content providers in a P2P network they do not loose control over their learning resources but still provide them for use within the network. As content consumers, both teachers and students, benefit from having access not only to a local repository, but to a whole network, using queries over the metadata distributed within the network to retrieve required resources.

Recent P2P applications have been very successful for special cases like exchanging audio files. However, retrieving MP3 coded audio files using title and author does not need complex query languages nor complex metadata, so special purpose formats for these P2P applications have been sufficient. Metadata in Gnutella are limited to a file name and a path. This is fine for queries looking for the song “Madonna - Like a Virgin”, but cannot be extended to something like “Introduction to Algebra - Lecture 23”. For educational resources, queries are more complex and have to build upon standards like IEEE-LOM/IMS [2] metadata with up to 100 metadata entries, which might even be complemented by domain specific extensions.

Furthermore, by concentrating on domain specific formats, current P2P implementations appear to be fragmenting into niche markets instead of developing unifying mechanisms for future P2P applications.

In order to facilitate interoperability and reusability of educational resources, we need to provide an infrastructure flexible enough to accommodate complex and varying metadata sets, and avoid creating another special purpose application suitable only for a specific application area which is outdated as soon as metadata requirements and definitions change. The Edutella infrastructure [4] therefore builds on the W3C metadata standard RDF(S) [1], and uses a standard query model suitable for this formalism, based on Datalog, to exchange queries throughout the Edutella network.

For the local user, the Edutella network transparently provides access to distributed information resources, and different clients/peers can be used to access, retrieve and update these resources. The service and the peer that we will describe in more detail in this paper are querying and annotating resources distributed in the Edutella P2P network, respectively.

Query Service. The Edutella query service is the most basic service within the Edutella network. Peers register queries they may be asked through the query service (i.e. by specifying sup-

Submission to Semantic Web Workshop 2002 Honolulu, Hawaii, May 7, 2002

ported metadata schemas (e.g. “this peer provides metadata according to the LOM 6.1 or DCMI standards”) or by specifying individual properties or even values for these properties (e.g. “this peer provides metadata of the form dc_title(X,Y)” or “this peer provides metadata of the form dc_title(X,’Artificial Intelligence’)”). Queries are sent through the Edutella network to the subset of peers who have registered with the service to be interested in this kind of query. The resulting RDF models are sent back to the requesting peer.

Edutella Annotation. In order to be able to meet the requirements of being applicable in a wide range of application scenarios, the Edutella annotation tool must be independent from a particular domain. For instance, it may not just offer annotation for IEEE LOM, but rather it must support a wide range of semantic definitions as it is possible in RDF schema. In order to approach this objective, we investigate two orthogonal dimensions for metadata creation based on which we may emulate all annotation schemes we know of.

Building on the JXTA P2P Framework. JXTA is an Open Source project [5] supported and managed by Sun Microsystems. In essence, JXTA is a set of XML based protocols to cover typical P2P functionality. It provides a Java binding offering a layered approach for creating P2P applications (core, services, applications). In addition to remote service access (such as offered by SOAP), JXTA provides additional P2P protocols and services, including peer discovery, peer groups, peer pipes, and peer monitors.

Figure 1, reproduced from [5], specifies the different layers within the JXTA architecture.

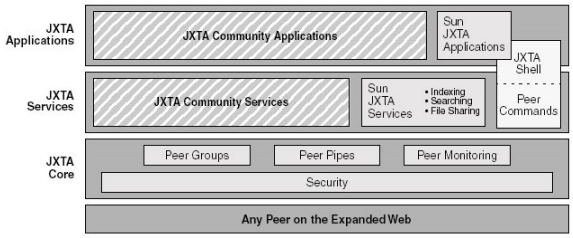

Figure 1: JXTA Layers

JXTA provides a layered architecture that fits very nicely into the Edutella application scenarios:

- Edutella Services (described in web service languages like DAML-S or WSDL, etc.) complement the JXTA Service Layer, building upon the JXTA Core Layer[1], and
- Edutella Peers live on the Application Layer, using the functionality provided by these Edutella services as well as possibly other JXTA services.

On the Edutella Service layer, we define data exchange formats and protocols (how to exchange queries, query results and other metadata between Edutella Peers), as well as APIs for advanced functionality in a library-like manner. Applications like repositories, annotation tools or GUI interfaces connected to and accessing the Edutella network are implemented on the application layer.

[1] A previous prototype from our group, implemented this summer to gain experiences with the JXTA and JXTAsearch framework, extended the JXTAsearch service (the prototype and our experiences with it are described in [13]), but building directly on the JXTA Core services makes a more flexible design possible.

In section 2, we discuss the Edutella query service and the common data model (ECDM), which provides the basis for the Edutella query exchange language and format implementing distributed queries over the Edutella network, as well as the basic Edutella API for query and registration/distribution peers. Section 3 discusses the annotation application that connects to the Edutella network in order to exploit existing metadata as well as create and provide new metadata to the network.

2. EDUTELLA QUERY SERVICE

2.1 The Query Mechanism

The Edutella Query Service is intended to be a standardized query exchange mechanism for RDF metadata stored in distributed RDF repositories and is meant to serve as both query interface for individual RDF repositories located at single Edutella peers as well as query interface for distributed queries spanning multiple RDF repositories. An RDF repository (or knowledge base) consists of RDF statements (or facts) and describes metadata according to arbitrary RDFS schemas.

To enable a peer to participate in its network, Edutella uses wrappers based on both a common datamodel and a common query exchange format. For communication within the Edutella network the wrapper translates the local data model into the Edutella common data model (ECDM) and vice versa, and connects to the Edutella Network using the JXTA P2P primitives, transmitting the queries based on the ECDM in RDF/XML form. In order to describe and handle different query capabilities of a particular peer, we define several RDF-QEL-i exchange language levels with increasing expressiveness: Currently we have defined language levels RDF-QEL-1, -2, -3, -4 and -5 (see [11]). The most simple language (RDF-QEL-1, purely conjunctive queries) can be expressed as unreified RDF graph, the more complex ones are more expressive than RDF itself, and therefore have to be expressed using reified RDF statements (e.g. RDF-QEL-3 covers relational algebra, RDF-QEL-4 incorporates Datalog). However, all language levels can be represented through the same internal ECDM data model.

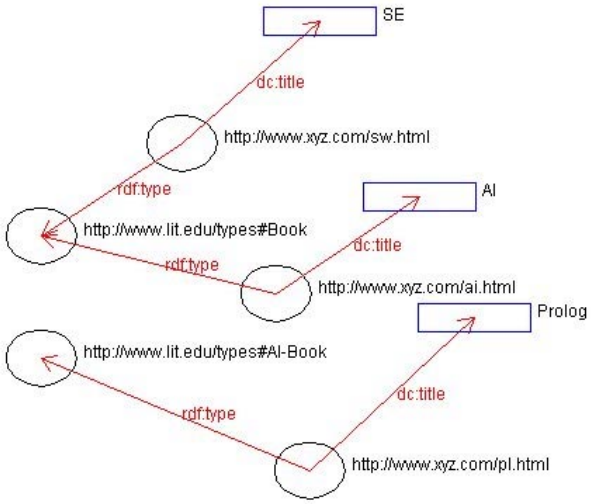

Figure 2: Knowledge Base as RDF Graph

The example presented throughout our paper, we will use a simple RDF knowledge base and a simple query on the knowledge base

depicted in Figure 2. Evaluating the query (plain English)

> "Return all resources that are a book having the title 'AI' or that are an AI book."

we get the query results shown in Figure 3, represented as an RDF-graph.

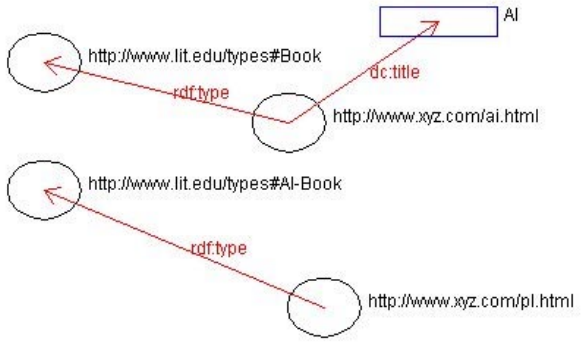

Figure 3: Query Results as RDF Graph

2.2 Edutella Common Data Model (ECDM)

2.2.1 Basic Semantics

As common query and datamodel, Edutella peers use Datalog, a non-procedural query language based on Horn clauses without function symbols. A Datalog program can be expressed as a set of rules/implications (where each rule consists of one positive literal in the consequent of the rule (the head), and one or more negative literals in the antecedent of the rule (the body)), a set of facts (single positive literals) and the actual query literals (a rule without head, i.e. one or more negative literals). Literals are predicate expressions describing relations between any combination of variables and constants such as title(http://www.xyz.com/book.html, 'Artificial Intelligence'). Disjunction in a query is expressed by a set of rules with identical head. A Datalog query then is a conjunction of query literals plus a possibly empty set of rules.

Datalog queries easily map to relations and relational query languages like SQL. In terms of relational algebra Datalog is capable of expressing selection, union, join and projection and hence is a relationally complete query language. Additional features include transitive closure and other recursive definitions.

In RDF any statement is considered to be an assertion. We can view an RDF repository as a set of ground assertions either using binary predicates as shown above, or as ternary statements "s(S,P,O)", if we include the predicate as an additional argument. In the following query, we use the binary predicate notation.

```
aibook(X) :- title(X, 'AI'), type(X, Book).
aibook(X) :- type(X, AI-Book).
?- aibook(X).
```

As our query is a disjunction of two conjunctive subqueries, its Datalog representation is composed of two rules with identical heads. The literals in the rules' bodies directly reflect RDF statements with their subjects being the variable X and their objects being bound to constant values such as 'AI'. Literals used in the head of rules denote derived predicates. In our example, the query expression "aibook(X)" asks for all bindings of X, which conform to the given Datalog rules and the knowledge base to be queried (cf. below for results).

2.2.2 ECDM Datamodel and Queries

Figure 4 visualizes the ECDM, as implemented in our current prototype, as UML diagram. Our Java binding relies on JXTA [5] and makes extensive use of the Stanford RDF API [10]. The implementation of all classes shown in figure 4 is found in the Java package `net.jxta.edutella.util.datamodel`. All classes whose names start with RDF represent standard RDF concepts and correspond to their equivalent counterparts within the Stanford RDF API. These are `RDFReifiedStatement`, `RDFNode`, `RDFResource`, `RDFLiteral` and `RDFModel`.

Queries are represented by `EduQuery` which aggregates an arbitrary number of rules (`EduRule`) and query literals (`EduLiteral`). `EduLiterals` are either `RDFReifiedStatements` (binary predicates / ternary statement literals, corresponding to reified RDF statements), `EduStatementLiterals` (non-ternary statement literals, which cannot be expressed as ordinary RDF statements), or `EduConditionLiterals` (a condition expression on variables such as $X > 5$).

Technically, it is sufficient to define a single instance of `EduLiteral` as query literal. However, by using a set of `EduLiteral` objects, all query literals together can be interpreted as the RDF result graph of the `EduQuery`, as long as the query literals are all instances of RDFReifiedStatement.

An `EduRule` consists of an `EduStatementLiteral` as its head and an arbitrary number of `EduLiterals` as its body. `EduStatementLiterals` can occur within a rule's body as well to allow reuse of other rules and recursion.[2]

`EduVariable` objects are ordinary RDF resources with the super class `RDFResource`. Being of type `EduVariable` however marks a resource to be a variable. An additional attribute allows to specify the label of a variable. Variables may occur in all places where `RDFResources` are allowed: As subject, predicate or object within `RDFReifiedStatements` as well as arguments of `EduStatementLiterals` or `EduConditionLiterals`. The class `EduVariableBinding` introduces a further extension to `EduVariable` by providing an actual value for a variable. Variable values can be either `RDFResource` or `RDFLiteral` objects.

Besides the ECDM data model the Java binding also provides a package `net.jxta.edutella.util` which contains classes for importing queries provided in various languages into the internal ECDM model or in turn exporting queries from their internal representation into different syntaxes. The current prototype includes the classes `SQL` (export of SQL queries), `Datalog`, `RDFQEL1` and `RDFQEL3` (all of them supporting import and export of queries). Any peer can plug in additional classes here to support further query languages (see [11]).

[2] Note, that as input format we can even allow arbitrary first order logic formulas in the body of rules, which then can be transformed into a set of rules using the Lloyd-Topor transformation [8].

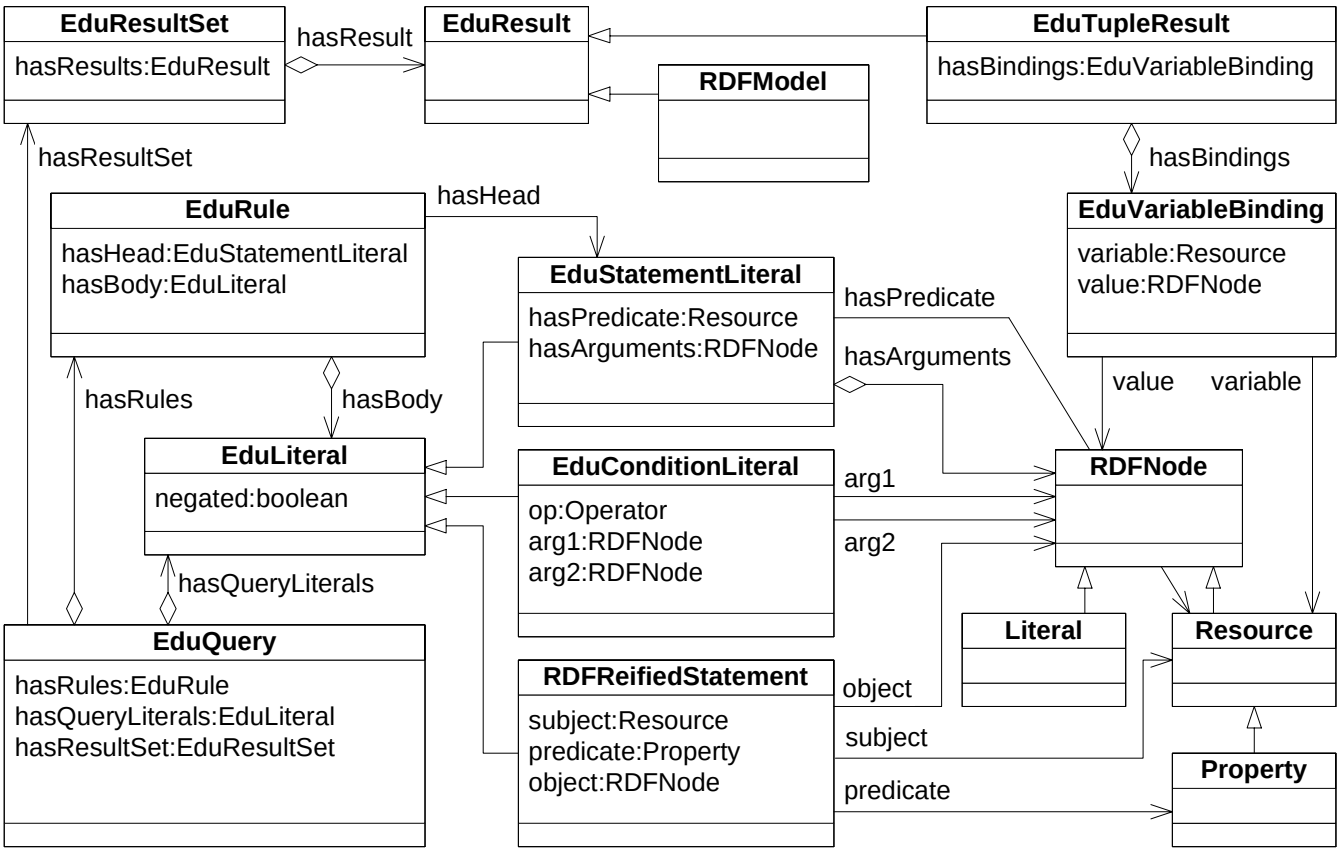

Figure 4: Edutella Common Data Model (ECDM)

2.2.3 Query Results

As a default, we represent query results as a set of tuples of variables with their bindings serialized in XML/RDF-format, as specified in Figure 4, which follows closely the convention of returning substitutions for variables occuring in queries to logic programs.

Another possibility, which has been explored recently in Web related languages focusing on querying semistructured data, is the ability to create objects as query results.

In the simple case of RDF-QEL-1, we can return as answer objects the graph representing the RDF-QEL-1 query itself with all Edutella specific statements removed and all variables instantiated. The results can be interpreted as the relevant sub graph of the RDF graph we are running our queries against (see Figure 3). When we use general RDF-QEL-i queries, we assume the structure of the answer graph to be defined by the subset of binary query literals. Note, that all variables used in the query literals are assumed to be existentially quantified, so if they are not instantiated during the query evaluation, they are represented as anonymous nodes in the RDF answer graph.

In the ECDM, `EduResult` is an abstract super class for different forms of query result representations. Results may be either represented as tuples (`EduTupleResult` objects aggregating an arbitrary number of `EduVariableBindings`) or as RDF graphs (`RDFModel` objects). In terms of relational algebra each `EduResult` object can be interpreted as one row in the result set of a relational database query. Each `EduResult` object corresponds to one match for a query. `EduResultSet` objects aggregate an arbitrary number of `EduResult` objects and represent a complete result set for an Edutella query. The individual results may be either `EduTupleResult` or `RDFModel` objects but they are all required to have the same type. When executing a query all query literals are evaluated using the necessary rules. After query execution a `EduQuery` object references an appropriate `EduResultSet` object pointing to all query results.

Classes within `net.jxta.edutella.util` also allow the import and export of query results in various other formats. Currently implemented are the classes `SQL` (for import of results provided as JDBC `ResultSets`) and `GraphViz` (for export of graph description files in GraphViz format allowing to use the free GraphViz tool to visualize query results).

2.3 Registration Service and Mediators

The *wrapper-mediator* approach introduced in [17], divides the functionality of a data integration system into two kinds of subsystems. The *wrappers* provide access to the data in the data sources using a *common data model* (CDM) and a *common query language*. The *mediators* provide coherent views of the data in the data sources by performing semantic reconciliation of the CDM data representations provided by the wrappers. Both common data model (ECDM) and common query language for the Edutella network have been defined in this paper.

Our simple “wrapping” mediators (see Figure 5) distribute queries to the appropriate peer with the restriction that queries can be answered completely by one Edutella peer. Complex ’integrating’ mediators are discussed in [11].

Registration of peer query capabilities is based on (instantiated) property statements and schema information, basically telling the

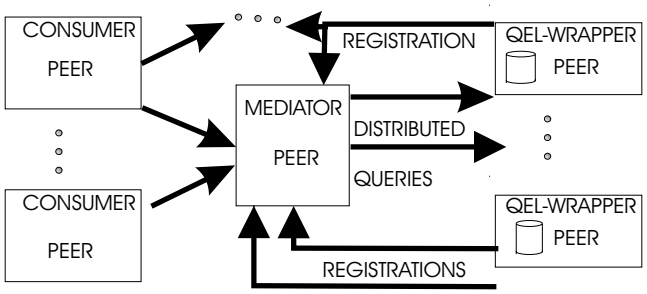

Figure 5: Query Mediator Wrapper

network, which kind of schema the peer uses, with some possible value constraints. These registration messages have the same syntax as RDF-QEL-1 queries, which are sent from the peer to the registration / query distribution hub. Additionally, the peer announces to the hub, which query level it can handle (RDF-QEL-1, RDF-QEL-2, etc.) Whenever the hub receives queries, it uses these registrations to forward queries to the appropriate peers, merges the results, and sends them back as one result set.

The packages `net.jxta.edutella.peer`, `net.jxta.edutella.provider`, `net.jxta.edutella.hub`, `net.jxta.edutella.consumer` contain interfaces to handle the distributed query mechanisms

Possible other registration methods would include specific term hierarchies which can be used as property value. A simple version could be **registerPropertyValue()**.

The query message contains not only the query itself but also information about query and result type (e.g. QEL-1, QEL-3 for queries and RDFModel, EduTupleResult for results). The returned message contains the original query in addition to its results.

3. EDUTELLA ANNOTATION

3.1 RDF(S) Annotation in a Nutshell

In order to easily provide metadata for a particular document, the annotation service provides a document viewer. Currently, the document viewer may display HTML pages, an extension for PDF documents is underway.

Furthermore, the annotation service provides a browser for RDF schema. This means that a corresponding definition, e.g. Dublin Core in RDFS[3] is loaded into the annotation tool and may be browsed. Fields for annotation are displayed according to the schema definition and may either be filled by typing or by marking and dragging information from the document viewer.

Thereby, annotations and fields for annotations may take quite a number of different guises. In our context an annotation is a set of instantiations attached to an HTML document. We distinguish (i) instantiations of RDFS classes, (ii) instantiated properties from one class instance to a datatype instance, and (iii) instantiated properties from one class instance to another class instance. Class instances have unique URIs. Instantiations may be attached to particular markups in the HTML documents, viz. URIs and attribute values may appear as strings in the HTML text.

For instance, one may decide, (i), to create an identifier for a person by instantiating HTTP://WWW.AIFB.UNI-KARLSRUHE.DE/WBS/SHA/#HANDSCHUH from the class DC:CREATOR and for a course HTTP://WWW.AIFB.UNI-KARLSRUHE.DE/LEHRVERANSTALTUNGEN/WINTER/EBIZ+INTELLIGENTWEB/#COURSE from the class SWRC:SEMINAR. (ii), one may instantiate the attributes of the first identifier by names like “Siegfried Handschuh” or “Siggi”. (iii), one may relate instances, e.g. the first with the second identifier by the property SWRC:TEACHES.

These types of instantiations may be considered one dimension in the metadata creation process. Another, orthogonal, dimension is defined by the way annotations are created, used and maintained:

1. *Unlinked facts* fill fields of the schema. There is no correspondence to the given document that is recognizable by the machine.

2. *Quotations* are excerpts from the document. E.g. a name like “Tim Berners-Lee” may appear in the document and also fill a field of the RDF schema description.

3. *References* are pointers to parts of the document. We use XPointer to select parts of the document. E.g. one may assert that a particular cell of a HTML table contains the name of the president of the U.S.A. — and in the right context one might expect that it is updated if it changes.

By the combination of these two dimensions (and the corresponding implications) we may emulate the metadata structure of all the different annotation tools that we currently know of (cf. [6] for a longer list of free and commercial tools).

3.2 Architecture

The Edutella annotation service is composed of the Edutella Peer structure and the KAON tool-suite [4] [9] incorporating the OntoMat Plugin Framework[5] and Annotation application [6] (cf. Figure 6).

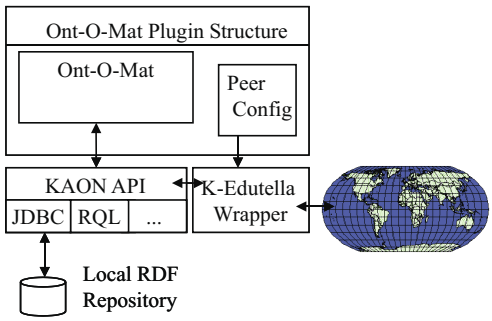

Figure 6: Ont-O-Mat as Edutella Peer

KAON is a Semantic Web tool suite originally created in isolation of Edutella. The OntoMat Framework is part of this tool suite and provides a java-based plug in structure which allows for loading services dynamically. One such service is Ont-O-Mat, which constitutes an annotation tool in the sense described above. Ont-O-Mat uses the KAON API to query for RDF schema definitions in order to build up its ontology browser. It queries for instances, attributes and relationships in order to let its users explore the current state of the knowledge base, e.g. in order to directly relate HTTP://WWW.AIFB.UNI-KARLSRUHE.DE/WBS/SHA/#HANDSCHUH with HTTP://WWW.AIFB.UNI-KARLSRUHE.DE/LEHRVERANSTALTUNGEN/WINTER/EBIZ+INTELLIGENTWEB/#COURSE.

The KAON API hides the actual implementation of the repository and the query language used. For instance, it allows to connect

[3] http://www.ukoln.ac.uk/metadata/resources/dc/datamodel/WD-dc-rdf/.

[4] http://kaon.semanticweb.org/

[5] http://annotation.semanticweb.org/

to a KAON RDF repository via a simple JDBC connection or to a RQL-based repository. The repository is used in two ways. First, it stores already available metadata and serves them to Ont-O-Mat via KAON API in order to allow for coherent metadata. This way, the chance is increased that there is only *one* identifier for the person named "Siegfried Handschuh" at Institute AIFB. Our experiences have shown that without such service several identifiers for single persons are created. Second, it stores metadata created by Ont-O-Mat.

Furthermore, we provide an Edutella Wrapper for KAON (K-Edutella Wrapper), which — like the corresponding Peer Configurator GUI — is a KAON plugin. The task of the K-Edutella Wrapper is to wrap the KAON-API for QEL and vice versa. The K-Edutella Wrapper calls JXTA lower levels for services like registration, pipes, etc. in order to connect to the outside world. Thus, from the point of view of the Ont-O-Mat user, he has a tool that may directly connect the Edutella network in order to query metadata from other peers or provide metadata from his repository.

4. CONCLUSION

Our prototype scenario features a set of (already existing) peers, which we have extended with the appropriate Edutella wrappers, and which connect to the Edutella framework with the functionalities *local and distributed queries* described in Section 2. The first prototype already contains the QEL query exchange mechanism, a simple mediatior and the wrapping of different repository peer types:

1. OLR (Open Learning Repository)[3] based peers using a subset of IMS/LOM metadata;
2. DbXML-based peers [14] as a prototype for an XML repository using a simple mapping service to translate from RDF-QEL-1 queries (conjunctive queries) to Xpath queries over the appropriate XML-LOM schema;
3. AMOS-II-based peers [15] with local repositories;
4. KAON-based peers [9] allowing remote annotation [6] using an RDF-based ontology format;
5. Concept-Base, a repository with full datalog capabilities [7].

Moreover the resulting environment will allow the design and integration of other tools which make use of metadata. In addition to the Ont-O-Mat, other applications such as Conzilla [12], which uses graphs to input queries and visualize results, will benefit from the Metadata enhanced Peer-to-Peer capabilities of Edutella. These first steps being done, a certain number of ameliorations are planned. In particular we will have to tackle mechanisms that provide replication of data implementing a a modification exchange language (MEL) and that resolve scalability issues like the selection of appropriate hubs for given queries.

Acknowledgements

The Edutella architecture, its query language and its various aspects have been defined in numerous discussions with many other participants in the Edutella project, and we gratefully acknowledge their support and important influence for this paper.

5. REFERENCES

[1] Dan Brickley and R. V. Guha. W3C Resource Description Framework (RDF) Schema Specification. http://www.w3.org/TR/1998/WD-rdf-schema/, March 2000. W3C Candidate Recommendation.

[2] IEEE Learning Technology Standards Committee. IEEE LOM Working Draft 6.1. http://ltsc.ieee.org/wg12/index.html, April 2001.

[3] Hadhami Dhraief, Wolfgang Nejdl, Boris Wolf, and Martin Wolpers. Open learning repositories and metadata modeling. In *International Semantic Web Working Symposium (SWWS)*, Stanford, CA, July 2001.

[4] The Edutella Project. http://edutella.jxta.org/.

[5] Li Gong. Project JXTA: A technology overview. Technical report, SUN Microsystems, April 2001. http://www.jxta.org/project/www/docs/TechOverview.pdf.

[6] Siegfried Handschuh and Steffen Staab. Authoring and annotating Web pages in CREAM. In *Proceedings of WWW-2002*. ACM Press, October 2002.

[7] M. Jarke, R. Gallersdörfer, M. Jeusfeld, M. Staudt, and S. Eherer. ConceptBase - a deductive object base for meta data management. *Journal on Intelligent Information Systems*, 4(2):167 – 192, 1995.

[8] J. W. Lloyd and R. W. Topor. Making prolog more expressive. *Journal of Logic Programming*, 3:225–240, 1984.

[9] Alexander Maedche, Steffen Staab, Rudi Studer, York Sure, and Raphael Volz. Seal — Tying up information integration and Web site management by ontologies. *IEEE Data Engineering Bulletin*, March 2002. http://www.research.microsoft.com/research/db/debull/.

[10] Sergey Melnik. *RDF API Draft*, January 2001. http://www-db.stanford.edu/ melnik/rdf/api.html.

[11] Wolfgang Nejdl, Boris Wolf, Changtao Qu, Stefan Decker, Michael Sintek, Ambjörn Naeve, Mikael Nilsson, Matthias Palmér, and Tore Risch. Edutella: A P2P networking infrastructure based on RDF. In *Proceedings of WWW-2002*. ACM Press, 2002.

[12] Mikael Nilsson and Matthias Palmér. Conzilla — towards a concept browser. Technical Report CID-53, TRITA-NA-D9911, Department of Numerical Analysis and Computing Science, KTH, Stockholm, 1999. http://kmr.nada.kth.se/papers/ConceptualBrowsing/cid_53.pdf.

[13] Changtao Qu and Wolfgang Nejdl. Exploring JXTAsearch for P2P learning resource discovery. Technical report, Learning Lab Lower Saxony, University of Hannover, November 2001.

[14] Changtao Qu and Wolfgang Nejdl. Towards interoperability and reusability of learning resources: A SCORM-conformant courseware for computer science education. Technical report, Learning Lab Lower Saxony, University of Hannover, October 2001.

[15] T. Risch and V. Josifovski. Distributed data integration by object-oriented mediator servers. *Concurrency and Computation: Practice and Experience*, 13(11):933 – 953, 2001.

[16] Steve Waterhouse, David Doolin, Gene Kan, and Yaroslav Faybishenko. Distributed search in p2p networks. *IEEE Internet Computing*, 6(1):68–72, January/February 2002. Special issue on Peer-to-Peer Networking.

[17] Gio Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38 – 49, 1992.

Smart Style on the Semantic Web*

Jacco van Ossenbruggen
CWI
P.O. Box 94079
1090 GB AMSTERDAM
The Netherlands
Jacco.van.Ossenbruggen@cwi.nl

Lynda Hardman
CWI
P.O. Box 94079
1090 GB AMSTERDAM
The Netherlands
Lynda.Hardman@cwi.nl

ABSTRACT

Web publishing systems have to take into account a plethora of Web-enabled devices, user preferences and abilities. Technologies generating these presentations will need to be explicitly aware of the context in which the information is being presented. Semantic Web technology can be a fundamental part of the solution to this problem by explicitly modeling the knowledge needed to adapt presentations to a specific delivery context. We propose the development of a *Smart Style* layer which is able to process metadata that describes content and use this metadata to improve the presentation of the content to human users. In the paper, we derive the requirements of such a Smart Style layer by considering Web design from both the document engineering and graphic design perspectives. In addition, design trade-offs made by human designers have to be taken into account for the automated process. After stating the requirements for a Smart Style layer, we discuss to what extent the currently available Web technology can be used and what its limitations are. The limitations are illustrated with examples of potential future extensions.

Keywords: Semantic Web, Device Independent Authoring, Document Engineering, Graphic Design.
Word count: 7150

1. INTRODUCTION

As the Web continues to grow not only in size but also in complexity, the increasingly varying needs of the intended audience marks the end of the "one size fits all" era. Delivery contexts [27] can be characterized in terms of specific user preferences and abilities, capabilities of the access device and available network resources. Given this heterogeneity, any single message needs to be adapted to a particular set of circumstances. As a minimum requirement, the author's intended message needs to be conveyed to the user given the constraints imposed by the access device. In addition, the generated presentation should conform as much as possible to the preferences of the user and the author [6]. These two types of adaptation may lead to an explosion of potential delivery contexts with which current stylesheet technology is unable to deal.

In previous work, we describe our prototype multimedia presentation generation system Cuypers [21]. Cuypers generates multimedia presentations adapted to the constraints of a specific delivery context. We claim that the particular solutions deployed within Cuypers realize a level of adaptivity that should become generally available on the Web. This introduces new challenges since the solutions need to be embedded within the current Web infrastructure. In this paper, we introduce the concept of *Smart Style*: an intelligent presentation adaptation layer for the Web that builds upon two fundamental technologies:

1. Web document engineering technology, including delivery formats such as HTML [30], SMIL [29], SVG [10] and XSL [28], and style and transformation languages such as CSS [4] and XSLT [7].

2. Semantic Web knowledge representation and metadata technology, including RDF [24], RDF Schema [25], DAML+OIL [19] and CC/PP [26].

Currently, Semantic Web technology is primarily deployed to improve Web-based information *gathering* and *brokerage*, with little attention to improving information *presentation*. Our vision is, however, that the Semantic Web infrastructure should not only play a key role in finding information on the Web, but also in presenting this information in the most appropriate way to each individual reader. Our proposed Smart Style layer will deploy Semantic Web technology to improve the presentation's adaptation, aiming for an optimized design of the presentation that suits the specific requirements of the user's delivery context.

In this paper, we derive the requirements for realizing the Smart Style layer. In section 2, we specify the key design ingredients of a Web-based presentation from two perspectives: a document engineering and a graphic design perspective. These allow internal trade-offs to be made in the design of a presentation. In addition, external forces that influence the decisions made during the design process are discussed in section 3. Both sections contribute to a set of requirements for Smart Style. Section 4 states to what extent the requirements are met in terms of the current Web infrastructure, identifies gaps and gives suggestions for extending the current Web infrastructure.

*This is a revised version of CWI Technical Report INS-R0201

Semantic Web Workshop 2002 Hawaii, USA

2. DESIGN PERSPECTIVES ON WEB PRESENTATIONS

In this section we compare two different perspectives of creating a presentation: document engineering and graphic design. The former assumes that the authoring process can be broken down into a sequence of sub-processes which are able to operate independently to generate the end result. The latter assumes a content-provider with a message to be communicated to a target audience, both of which the designer has to understand exactly before creating the appropriate mix of graphics and text to effectively communicate the client's message. Both perspectives are valid and need to be understood before distilling the requirements for a Smart Style layer.

2.1 Document Engineering Perspective

From a document engineering perspective, it is important to separate *content* from *style* information. The underlying principle is that the essence of the message is contained in the (XML-structured) text and remains unchanged when style parameters, such as screen width or font size, are varied. This principle allows the creation of an infrastructure where the file containing the content, the XML file, can be created and maintained separately from a style file, such as a CSS stylesheet. The advantages of this approach are well-known within the Web community. These include the reuse of the same content in different contexts and the enforcement of consistent styles across different sets of content [20].

A presentation, however, involves more than applying an appropriate style to the selected content. A third, and essential, ingredient is the *structure* of the presentation. The simple separation of content and style as described above suffices only when the presentation structure is similar to the content structure in the underlying XML. If this is not the case, then a transformation step, such as enabled by XSLT, is needed to convert the content structure to the desired presentation structure. For example, the lexical order in a source HTML document might need to be transformed to the order that is most appropriate in the text-flow of the target HTML presentation. Alternatively, a more structural process may be needed, such as a transformation of an XML document into an XSL formatting object tree.

The document engineering process of creating Web presentations can be summarized in three steps:

1. select or create the content (typically structured using XML);
2. define a mapping from content to the presentation structure that defines, among other things, the most appropriate order (e.g. by using XSLT);
3. (optionally) refine this presentation structure by applying preferred style parameters (e.g. by using CSS).

Essential in this approach is the assumption that the three steps can be carried out independently. Content can be entered into a database by a content-provider. This content can then be extracted from the database in the desired order by a server-side script written by a Web-site programmer. Finally, the preferred style parameters can be determined (server-side) by a graphic designer's and/or (client-side) by the end-user's stylesheet. For many (database) content-driven Web sites, this assumption holds. The same applies to knowledge-driven or model-driven sites (See for example, [12]). Furthermore, the current Web infrastructure, with its large number of XML-related tools, is well equipped to support this process.

2.2 Graphic Design Perspective

Despite the advantages of the document engineering approach, it also has significant limitations. Specifically, in our own work on automatically adapting multimedia presentations to a variety of delivery contexts, generic XML tools proved to be inadequate (see [21] for details). Current tools are unable to deal with multimedia content for which it is not known *a priori* which transformation and stylesheet are suitable for displaying the content in a particular context. In online multimedia databases, for example, multimedia presentations can be generated from the media items returned by a database query. Since information about the media items such as quantity, type, size and size is not known in advance, template-based solutions cannot be used for determining a suitable presentation structure. Several solutions for this problem have been proposed and include: the use of large numbers of templates, where selection of the correct template becomes a problem [9]; constraint-based approaches, using grammars [31], planners [1] or logic programming [21] to generate the constraints; and other model-driven approaches to automatic presentation generation [3, 13].

Presentation structure plays a much more important role in multimedia than in text-based applications. Multimedia users experience presentation structure primarily as the sequential arrangement of the constituent media items in time, as the spatial arrangement on the screen and as paths of navigational hyperlinks. The presentation structure of multimedia is more difficult to determine automatically by stylesheets. For text, stylesheets may change the layout (e.g. switch from single column to two column, or change the margins) while preserving the semantic integrity of the underlying message. For multimedia, changes in the spatio-temporal arrangement will often have a large impact on the perceived semantics of the presentation [22]. Multimedia formats such as SMIL [29] address this problem by allowing the author to specify the presentation structure explicitly in the document. This is required because in multimedia the message is conveyed not only by the individual media items but also by the spatio-temporal and navigational arrangements of the media items in the presentation. In multimedia the presentation structure and content are in general *not* independent.

The document engineering approach thus needs to be refined for media-centric applications, in which the assumption that content, presentation structure and style are independent is false. In contrast to the content-centric approach in most of the document engineering literature, most of the graphic design literature features a more balanced perspective on the relation between content, presentation structure and style and the roles these three ingredients all play in conveying the overall message of the presentation. Understanding these roles and their dependencies is crucial for determining the requirements of a Smart Style layer.

Figure 1 illustrates how decisions made in any of the three sections can influence the other two. We give examples of how each part of the figure influences the other two parts.

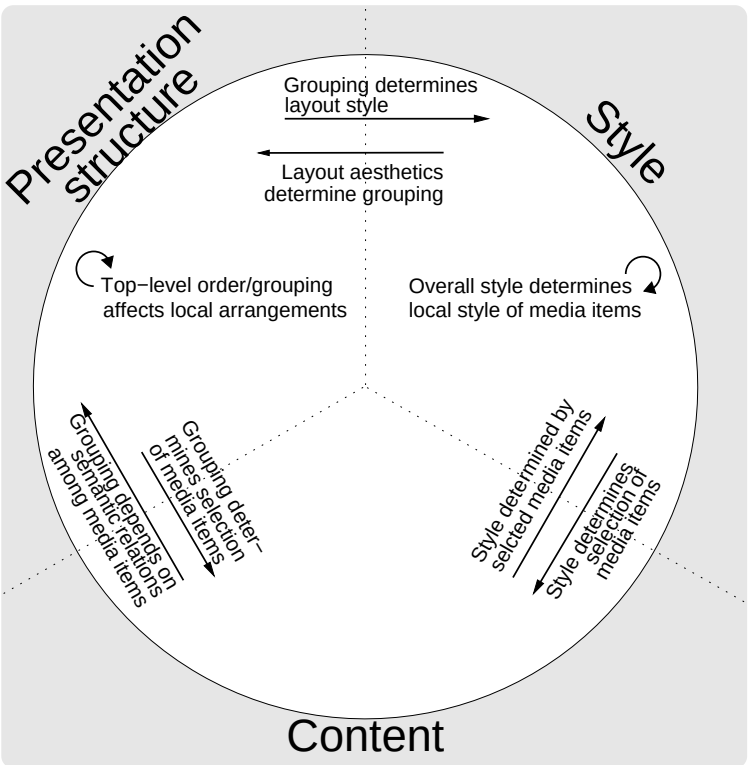

Figure 1: Dependencies between content, presentation structure, and style.

Presentation structure depends on content. The particular selection of content items can be used to determine the presentation structure of the items. For example, suppose that a number of media items have been selected for presentation. The items fall into three categories, and the presentation structure reflects these categories by first displaying all items in a single category before displaying the items from another category. For printed graphics, Williams [32] advocates the use of spatial layout to express grouping relationships in the underlying content. In multimedia time, space and links can all be used to communicate the underlying grouping relationships [18].

Style depends on content. The content can also influence the overall style of the presentation. For example, suppose a number of images are selected to convey the message. They happen to share a number of color characteristics, which lead to the choice of particular colors for the background and main text colors of the presentation. Other aspects, such as image texture, could, for example, also influence the selection of appropriate font type faces. Availability of appropriate content may influence the choice of style through, e.g., using an image as a background for the rest of the presentation.

Content depends on presentation structure. In the document engineering perspective, presentation structure is derived from the original content structure. In practice, however, when a Web site is created, the presentation structure, in particular the spatial layout and navigation structure among sections, is often determined first and the content created to fit into it. It is difficult, however, to make these dependencies explicit.

Style depends on presentation structure. The style can also depend on the presentation structure. For example, if the presentation structure uses spatial alignment for conveying grouping relationships in the content, then the designer needs to choose a particular alignment style (e.g. left-, centered or right-aligned). In figure 2, for example, the title is centered above the text to convey the grouping relation, in this case that the title applies both to the explanatory text and to the example image. Note that the same presentation structure could have been conveyed using a longer title aligned left with the *chiaroscuro* explanation.

Content depends on style. In the document engineering perspective, style is often perceived as the "add-on" after the "important" decisions have been made. The "more fundamental" choice of content may, however, also depend on the style. For example, in order to preserve the visual unity in a presentation, relevant images may be selected for presentation only if their color histograms or clarity of images fit in with the style of the presentation as a whole [17]. For Web-sites aiming for a strong visual effect (e.g. for branding purposes), the look-and-feel of the site is typically designed first. The content is selected, manipulated or created to fit in with the chosen style.

Presentation structure depends on style. Similarly, an established style may prescribe certain limits on the spatial grid and pacing of the presentation. Ideal groupings and orderings of selected content may have to be put aside for reasonable alternatives which fit in with consistent use of margin widths and item alignments throughout the presentation. Similar effects and tensions are possible for temporal layout. A rhythmic presentation gives a certain desired effect but may clash with specific durations needed to express the message at different points in the presentation.

. In addition to the mutual dependencies between each of the three aspects, local presentation structure or style can depend on more global presentation structure or style. This can be used to provide continuity and consistency throughout the presentation.

In summary, in the graphic design perspective of creating a presentation, aspects of content, presentation structure and style depend on each other in ways that are generally ignored in the document engineering perspective. This is not to say that document engineering tools are not useful, but rather that the extra dependencies which make the task of good design so complex require more complex solutions. Since automated adaptation requires finding solutions within this design space, the three aspects of content, presentation structure and style need to be expressed and manipulated explicitly.

Being aware of the internal mutual dependencies is necessary but insufficient for finding a satisfactory solution in the design space. The adaptation also needs to fulfill external requirements of, e.g., the user and content provider. In the following section we discuss potential external influences on the design process.

3. EXTERNAL FORCES ON THE DESIGN PROCESS

The previous section explains the mutual dependencies that play a role when making decisions about content, presentation structure and style. So far, we have limited the discussion to the dependencies that are *internal* to the process of designing a Web presentation. In this section we discuss the dependency of design decisions on *external* forces.

The external forces that influence the design originate directly from the different interests of the parties involved. To determine the requirements of an automated system, we use the following motivating example, based on a typical scenario with three main parties: a *content-provider* who wishes to effectively communicate a message to a *user*, aided by a skilled *designer*.

Examples of forces that originate from the content provider include the *mission* of the content provider's organization (e.g. making profit by selling books online), the limited availability of *resources* (e.g. the amount of time and money the organization is willing to spend on the design, the amount of disk space or bandwidth that is available at the server), and the content provider's *preferences* (e.g. the use of company colors in the Web forms).

Examples of forces that originate from the user include the user's *needs* (e.g. the desire to buy a book), the limitations imposed by the user's *delivery context* (e.g. the user could be driving a car, have a low bandwidth connection, have physical disabilities, or have strict time constraints), and the user's personal *preferences* (e.g. user could prefer visual to textual information, dislike fast cuts in video material, prefer soft colors to primary colors).

Given a good understanding of the type of forces that play a role, it is the task of the designer to come up with a design that best matches the needs of the content provider and the user. In addition to the forces originating from the content-provider and the user, there are additional forces originating from the designer, whose resources are also limited and might also have personal preferences. Many of these forces could give rise to conflicts and will require the designer to make balanced trade-offs. For example, the designer might decide *not* to use the soft colors of the organization's company logo for users that need to fill in Web-forms while working in bad lighting conditions.

The role of an intelligent automatic adaptation mechanism is very similar to that of the human designer. Automatic adaptation also has to deal with forces originating from content-provider and user, as well as with forces originating from the adaptation process itself (e.g. limited computing resources, or personal preferences of the developer of the adaptation system). For example, in figure 2 the preferred design centers the title across the width of the screen. In figure 3, however, the client's platform display is shorter and the images are not to be scaled — a condition imposed by the copyright holder to preserve image quality. This forced our Cuypers system to search for an acceptable layout design alternative within the given device and content-provider constraints. We do *not* claim that we can build an automated system that could make such decisions as well as a professional designer. Intelligent adaptation systems such as Cuypers can, however, make acceptable design decisions when dealing with these types of trade-offs. Their intelligence is based on explicit knowledge about the design space dependencies and external constraints, combined with an adequate search strategy. These characteristics require that adaptation be more than the application of a simple mapping from source to destination format. Rather, it requires heuristic reasoning to find an optimal solution to balance the forces involved.

The requirements for automated adaptation following from the discussions here and in section 2 can be stated as follows.

- Explicit knowledge reflecting the internal design dependencies among content, presentation structure and style discussed in section 2, and external influences of the delivery context specifying the user's resources, preferences and needs, and the content provider's server context.

- A transformation method which can take the above knowledge into account, to make an informed choice in the internal design space while balancing the external trade-offs. Note that this transformation method will not be based on simple mappings.

In the next section we discuss the implications of these requirements for extending the current Web infrastructure.

4. TOWARDS INTELLIGENT STYLESHEETS

The type of adaptation that can be found on the Web today may seem to be a far cry from the type of intelligent adaptation discussed in the previous sections. To a certain

Figure 2: Example SMIL presentation (generated by Cuypers for display on large screen).

extent, however, the current Web infrastructure already provides a good basis upon which a smarter adaptation layer could be built. Making this layer work in practice, however, requires the specification of new standards, and — arguably more difficult — a sophisticated and seamless embedding of these new techniques in the current Web framework.

In this section, we give a short overview of how current Web standards relate to automatic adaptation. We then discuss the requirements of a Smart Style layer illustrated by examples of potential extensions of current Web technology. We use the current Web Recommendations as a basis, and incorporate, as much as possible, other W3C activity that is still work in progress.

4.1 Current Web adaptation techniques

To a certain extent, a Web site can already build its own server-side adaptation techniques by deploying generic Web technology, such as CSS and XSLT stylesheets, which can be used to adapt and style XML and HTML content. These techniques can be combined with commonly used solutions such as filling Web templates with material stored in databases.

Today's Web formats also allow client-side adaptation. Many delivery formats, including HTML, feature basic functionality to improve accessibility. A well-known example is the `alt` tag that can be used to provide an alternative, textual description of the role of an image within an HTML page. SMIL features a more sophisticated example in the `switch` element, which can be used by multimedia authors to provide alternatives for parts of the presentation depending on the delivery context. The XSL vocabulary [28] includes features that allow similar client-side adaptation, including the `role` property and the `multi-switch` formatting object. The use of metadata also has a huge potential for improving automatic adaptation. A good example is the work on automatic linearization of SVG documents, to allow synthesized speech browsing of SVG [16, 11].

The appropriate use of the techniques sketched above already requires human designers to deal with the design dependencies discussed in section 2 and trade-offs discussed in section 3. For example, multimedia authors have to make trade-offs between function and resources for each `switch` element in their SMIL presentations, because they need to match the functionality of their media items against the available bandwidth and others resources that are required by these media. The trade-offs between function and preferences in the company color scheme example are similar to the decisions made when using CSS `!important` rules, where designers need to think to what extent the preferences of the end-user's stylesheet are allowed to override the defaults determined in the document's stylesheet.

These current techniques suffice for applications in which a human designer has, during authoring time, sufficient information for making the required design decisions and trade-offs. For applications for which this information is only available at the time of the user request, the decisions need to be made by the adaptation system. The current Web infrastructure is, however, insufficient for making run-time design decisions. In order to move towards the intelligent adaptation and styling advocated in this paper, we need to extend the current Web framework.

4.2 Communicating delivery contexts

The first ingredient we need is a commonly agreed upon way to communicate the information upon which we will base our design decisions. A key requirement is the ability to communicate delivery contexts. Delivery contexts are required in order to provide information about the client-side

Figure 3: Same SMIL presentation, but adapted to the smaller height of the user's display.

resources that are available and about the personal preferences of the user. Assuming that at least a part of the adaptation will need to take place on the server, it is *essential* to standardize the communication of delivery contexts: clients need to be able to send the information in a way that the server understands. A machine-readable description of a delivery context that can be sent to the server is often called a *profile*. Within W3C, work on a common ground for delivery contexts is currently in progress. CC/PP [26] provides an RDF-based framework for defining the vocabularies that are needed to define profiles. In addition, it also provides a small vocabulary that can be reused across different profiles. A typical example of a CC/PP profile is the User Agent Profile developed by the WAP Forum [33]. This profile provides a commonly agreed upon mechanism to communicate the (technical) capabilities of mobile phones to servers and proxies. The CC/PP framework, however, is sufficiently flexible to allow the definition of profiles that focus on more user-centered aspects of a delivery context.

From a technical point of view, CC/PP is built on top of RDF. CC/PP profiles use RDF statements to describe the relevant client-side capabilities and preferences. For example, figure 4 shows a fragment of a delivery context that uses CC/PP to inform the server that the client platform features a 640x480 display.

CC/PP profiles are, at the time of writing, hardly used on the Web (the WAP industry forms a notable exception). Communicating delivery contexts between client and server needs to become standard practice, which is more than an implementation issue. Additional CC/PP vocabularies need to be provided, not only to describe the capabilities of the hardware and software of the user's device, but also to describe the needs, environment and personal preferences of the user.

```
...
<ccpp:component>
  <rdf:Description rdf:about="TerminalHardware">
      <rdf:type rdf:resource="HardwarePlatform" />
      <ccpp:pix-x>640</ccpp:pix-x>
      <ccpp:pix-y>480</ccpp:pix-y>
      ...
  </rdf:Description>
...
</ccpp:component>
```

Figure 4: Example fragment of a delivery context specified using CC/PP.

4.3 Supporting metadata for content description

Clients need to be able to communicate delivery contexts, but in itself this is insufficient. Many design decisions will also depend on information that is available at the server-side. Even when this information is not intended to be published on the Web, having commonly used and standardized solutions for describing and processing it will greatly reduce the development effort needed to implement a smart, adaptive Web site.

Intelligent adaptation systems will need some knowledge of the function of the content they are adapting. To make this type of knowledge explicit, appropriate use of *metadata* will be of key importance. Within and outside W3C, a large amount of work on metadata standardization is currently in progress, and in most of this work RDF plays a central role. For example, work on RDF Schema aims at adding functionality that allows RDF vocabularies to be defined in a standardized way. Ontology languages, such as DAML+OIL, built on top of RDFS, add features while still allowing efficient implementations that are able to reason

about metadata information.

While the current focus of this type of Semantic Web technology is on the use of metadata to achieve a more intelligent model for Web-based information retrieval (e.g. improving search engines), the use of metadata in our Cuypers system shows that there is also a huge potential in applying this type of technology for improving the adaptation and presentation process. Through the use of metadata to make the intended semantics and function of the content explicit, adaptation systems are able to make informed decisions during the design process. For example, suppose an online museum site has developed an RDF Schema[1] for the metadata[2] used to annotate their Web site. Also suppose the site features an HTML page describing a work by the painter Rembrandt van Rijn, focusing on the use of *chiaroscuro* (the painting technique that uses strong contrasts of light and dark paintings). Figure 5 shows a fragment of the HTML version of the earlier SMIL presentation.

```
<div id="allegory">
 <h1>Musical Allegory</h1>
 <img src="allegory.jpg"/ >
 <p>This is hardly just an ordinary group of musicians.
    The figures are too exotically dressed in oriental
    ...
</div>
```

Figure 5: Example XHTML 1.0 fragment from a page about a Rembrandt painting.

From an XML markup perspective, all we know is that we have a fragment with a first level heading, an image and a text paragraph. The underlying semantics, however, could be explicitly added by the use of RDF metadata, as shown in figure 6.

```
<museum:Painter rdf:ID="Rembrandt">
  <museum:fname>Rembrandt</museum:fname>
  <museum:lname>Harmenszoon van Rijn</museum:lname>
  <museum:painted rdf:resource="#allegory" />
</museum:Painter>

<museum:Painting rdf:about="#allegory">
   <museum:title>Musical Allegory</museum:title>
   <museum:technique>Chiaroscuro</museum:technique>
</museum:Painting>
```

Figure 6: RDF metadata of XHTML 1.0 fragment.

This explicitly states that our HTML fragment is an instance of a class `Painting`, with a `title` property "Musical Allegory", and that there is a `Painter` instance that has a `painted` relation with the painting at hand.

Given such semantic information about the content, and the explicit descriptions of the delivery context, adaptation engines should be able to make better decisions about how to adapt the presentation to a particular situation. For example, because the metadata explicitly states that the painting is using the technique "Chiaroscuro", an adaptation engine might decide to add, for non-expert users, a link to the page describing this technique. This requires an adaptation process that takes into account both the delivery context (because it needs to know that the user is a non-expert) and metadata (because it needs to know in which conditions it should add a link). Based on our experience with Cuypers, we found that most metadata is used for content descriptions that are defined in terms of the application domain. This may be sufficient for most information retrieval purposes, but not for information presentation. Metadata that, for example, identifies the potential role the content could play in the presentation is hard to find. In the example above it was hard to predict on the basis of the metadata whether the textual description of the "Chiaroscuro" technique is suitable for non-expert users or not. For the images, it was hard to determine to what extent images could be resized to fit the presentation without compromising the information that was intended to be conveyed.

In general, to improve intelligent adaptation and presentation, metadata annotations of Web content is required. Annotation should, however, not be confined to information retrieval, but also facilitate information presentation.

4.4 Processing delivery contexts and content descriptions

Assuming that the information upon which we base our design decisions will be available from the Web through the use of standard Semantic Web technologies such as CC/PP and RDF, the next ingredient needed for building a Smart Style layer is an efficient set of tools that allows this information to be taken into account during the adaptation process. As described above, many of the current generation W3C Recommendations already have some features that address adaptation issues. A first step is to make the current generation presentation-oriented Web technology interoperable with the next-generation Semantic Web technology. For example, CSS stylesheets are currently not able to take CC/PP profiles into account. CSS has, however, a feature that is closely related to CC/PP, and allows the specification of device dependent style rules: the `@media` rule. Figure 7 shows an example[3] of a stylesheet that uses bigger fonts on computer screens than on paper printouts of the same document.

```
@media print {
  body { font-size: 10pt }
}
@media screen {
  body { font-size: 12pt }
}
```

Figure 7: Device dependent style rules as already supported in CSS2.

A first step towards a CSS syntax that allows more detailed queries is suggested in [15]. In this syntax, queries to specific device features are allowed. For example, the CSS media rule for screen display above could be further refined by adding constraints on the minimum width of the screen,

[1]Museum schema example adapted from [14].
[2]Metadata example adapted from [23]).
[3]Example taken from the CSS2 Specification [4].

as shown in figure 8. Using the constraints, stylesheets could take into account the information provided by profiles such as the example in figure 4.

```
@media screen and (min-width: 640px) {
  body { font-size: 14pt }
}
@media screen and (min-width: 800px) {
  body { font-size: 16pt }
}
```

Figure 8: Detailed media queries using a CSS3 extension (work in progress).

Even from this extended CSS syntax, however, it is still a long way to fully CC/PP aware style engines. CC/PP features that will affect style application include the ability to define new profile vocabularies, inheritance mechanisms for specifying default values and the description of the capabilities of transcoding proxies. Style engines need to be able to deal with these features in order to take full advantage of the information specified in CC/PP delivery contexts.

Note that the need to take CC/PP information into account also applies to XSLT transformation engines. While the full details of how this could affect future versions of XSLT are beyond the scope of this paper, one could, for example, imagine an extension[4] of XSLT's *mode* concept. For example, transformation rules could be selected in a way similar to that of the media rules in CSS. In such a hypothetical extension (see figure 9) one could, for instance, define a rule for creating a two column layout only if the output medium is print and the paper is wider than 17cm.

```
<xsl:template match="body"
              mode="print and (min-width: 17cm)">
  ...
  <fo:region-body column-count="2"/>
  ...
</xsl:template>
```

Figure 9: Device dependent rules by extending XSLT modes (tentative syntax).

In addition to taking information about delivery contexts into account, stylesheets also need to take into account the semantic information that is contained in the metadata associated with the content. Currently, style selector mechanisms only match on the *syntactic* properties of the underlying (XML) document hierarchy. This applies both to the selector mechanism used by CSS and to the XPath [8] selectors used by XSLT.

In all examples above, the rules were intended to match on the `<body>` element of an HTML document. Similar rules could be written to match on the syntactic properties of metadata, e.g. on the XML element and attribute names that are used to encode the RDF statements in figure 6. Using the current generation CSS and XSLT engines to process general metadata it is, however, not practical to match on the *semantic* properties of metadata: for CSS and XSLT processors, RDF is just XML. As a result, it is very hard to write, for example, a rule that matches on all alternative XML serializations that are allowed for RDF. A more serious problem, however, is that it is impossible to write CSS or XSLT rules that make use of the structural relations of RDF and RDF Schema, for instance a style rule that applies to all objects that are instances of a specific RDFS (sub)class. Neither is it possible to write rules for all objects that have a certain DAML+OIL-defined ontological relation, etc. Model-driven Web site management systems such as OntoWebber [12] are thus forced to develop their own solutions to associate presentation design elements to their RDF (and DAML+OIL) data, because CSS and XSLT are currently not applicable to RDF.

Future, Semantic Web-aware, selector mechanisms could allow specification of style rules in terms of the RDF semantics expressed in the metadata. This would extend the currently used CSS and XPath selectors, that are based on the XML syntax encoding the semantics. Consider the extended XSLT example rule in figure 10, which uses the RDF-aware query language RQL [14] for its selector, instead of XPath.

```
<xsl:template match=
  "RQL(http://www.museum.com/schema.rdf#Artifact)">
  ...
</xsl:template>
```

Figure 10: Semantic matching of XSLT rules using RQL selectors (tentative syntax).

The RQL query used would match on all instances of `Artifact` and its subclasses. Since our Museum RDF Schema defines `Painting` as a subclass of `Artifact`, the rule above would match on the semantics and structure of the RDF metadata describing the painting shown in figure 6, irrespective of the XML serialization syntax used to encode these semantics [5].

4.5 Beyond CSS and XSLT style and transformation rules

Above, we suggested extensions to CSS and XSLT that would allow stylesheets to take into account delivery contexts as specified by CC/PP and content semantics as expressed by RDF metadata. While taking this type information into account is a prerequisite for a Smart Style layer, this is in itself not sufficient.

Adaptation engines need to be able to search in the design space sketched in section 2, and make the trade-offs discussed in section 3. This type of decision process is hard to define using the simple "if *selector matches* then *apply rule body*" type of current style and transformation rules.

In addition, our experiments with the Cuypers system [21] allowed us to analyze the adaptation process of multimedia presentations for which the quantity, type and size of the media items were not known until run-time. We found that for these applications, automatic adaptation also requires the ability to verify the presentations that result from applying a set of transformation rules. When designing transforma-

[4]We are not advocating a specific syntax, but are only claiming that future XSLT transformations need to be able to take CC/PP-like information into account

tion rules for dynamic multimedia, one cannot, at authoring time, guarantee that the resulting presentation indeed meets the "hard" constraints imposed by the available resources. We have used the Cuypers system to experiment with a transformation engine that can evaluate the multimedia presentations it generates. The system employs backtracking to search for alternative rules when the end result does not meet the constraints imposed by the available resources. For example, even when a specific rule is applied only for target screens with a certain width, that condition in itself will not guarantee that the presentation resulting from applying the rule to media content of unknown size will indeed meet the maximum width constraints. What is needed is a means of evaluating the actual width of the final presentation, and a means of trying alternative rules in case the presentation did not meet the constraints.

While CSS and XSLT rules cannot be used to specify the required search strategies, this type of processing is vital for intelligent adaptive behavior on the Web. The Web thus requires more sophisticated ways of transforming the combined information provided by delivery contexts, metadata and the content into meaningful presentations. In future research, we want to explore how, and to what extent the combined search, transformation and evaluation techniques used within our Cuypers system could be made generally available on the Web.

4.6 Beyond atomic style properties

In addition to improved transformation processes, we also need to develop better abstractions to reason about the "soft" constraints imposed by the preferences of the parties involved. This type of reasoning requires explicit knowledge of the dependencies discussed in section 2. Taking these preferences and the associated dependencies into account will have a large impact on the perceived overall quality and design of automatic Web presentations. Currently, style rules work only on the basis of individual style properties. For example, one can specify the font type or color of a specific XML element. To what extent the application of these individual rules yield the desired overall result is hard to predict in advance, especially when dealing with more complex publishing systems that feature dynamic content, XSLT transformations, transcoding proxies and CSS stylesheets. After this process, the font and color of two XML elements positioned together in the final presentation might not go well together. Within the graphic design profession, style guidelines and checklists have been developed that can be used to avoid such design mistakes (see, for example, [32, 17]). It should be possible to build on this body of knowledge, and at least check the overall presentation against the most common design flaws. In addition to graphic design, similar checks could be developed for checking the design of the overall temporal flow of, and synchronization within, the presentation [2], and for checking the design of the navigation and interaction schemes that the presentation exposes to the end user.

5. CONCLUSIONS

Current Web technology addresses the problem of multiple delivery contexts through the use of CSS and XSLT stylesheets. These can be used for transforming presentation-independent XML documents to specific presentation formats, such as XHTML, SVG or SMIL. This is sufficient for dealing with a limited number of delivery contexts per stylesheet but is inadequate for adapting content to the plethora of delivery contexts for different devices, network resources and user groups. To solve this problem the Web is currently missing three key ingredients.

1. **Common vocabularies for describing delivery contexts** Web applications need to be able to communicate their capabilities and the preferences of their users so that transformation engines are able to make informed choices during the presentation generation or transcoding process. CC/PP already provides a framework for defining such vocabularies. Commonly agreed upon vocabularies will be needed for defining user preferences, device capabilities, network characteristics etc.

2. **Intelligent transformation methods** Transformations need to be able to take into account a wide variety of delivery contexts to generate a presentation corresponding to a particular delivery context. While it is unrealistic to expect that even an intelligent stylesheet would be sufficiently powerful to cater for *any* given delivery context, our claim is that the current transformation technologies can be significantly improved in order to allow a substantial increase in flexibility.

3. **Explicit metadata and design knowledge** Given the vocabularies for describing delivery contexts, and given an appropriate transformation method, in theory we would be able to develop adequate intelligent stylesheets. In practice, however, these stylesheets would implicitly contain a large amount of design knowledge and domain knowledge. This type of knowledge should preferably be made explicit and specified declaratively, in a similar manner to the explicit and declarative delivery contexts. RDF Schema [25] and DAML+OIL [19] already provide a framework for encoding this type of knowledge. To what extent vocabularies for this type of knowledge can be standardized remains to be seen, since they may be highly domain and application specific.

As these three ingredients build directly upon Semantic Web technology, we believe that only by a synthesis of (future) Semantic Web tools with the presentation-oriented tools of the (current) Web, can we hope to address the adaptation problems discussed.

This brings us to the first Achilles' heel of our Smart Style layer: the large amount of current and future W3C Recommendations that currently exist. Many of the Recommendations can be used to address part of the problem, but it is not clear how they can be used in concert to solve the overall problem. This paper derives the requirements for an ambitious goal: automatic adaptation of dynamic text and multimedia content to the requirements of the individual user's delivery context, while respecting the integrity of the semantics of the content. If we reduce our ambition levels, however, and "only" aim for taking into account processing context information, this alone would still have major consequences. To prevent CC/PP from becoming a standalone W3C recommendation that can only be processed with proprietary tools, we need to clearly define how other recommendations, including CSS, XSLT, XHTML, SMIL and SVG operate in the context of CC/PP. From CC/PP-aware

Web transformations, another step is required towards Semantic Web-aware transformations that also take metadata semantics into account. This will require tools that can abstract from the underlying XML syntax and operate directly on the semantics of languages such as RDF, RDFS and DAML+OIL. Realizing this level of interoperability among W3C Recommendations will be a huge effort. It should be clear that the examples given in this paper serve only to illustrate the derived requirements, and should by no means be regarded as readily applicable solutions to achieve the required interoperability. Making the current Web infrastructure interoperate seamlessly with the upcoming Semantic Web will be a huge challenge and a long term effort.

Finally, the other Achilles' heel of our Smart Style layer is the large amount of high quality design and domain knowledge that it requires. Smart Style does not aim at replacing human designers, but strives for providing applications with sufficient design knowledge when design decisions cannot be make by humans. It will require a large amount of human effort to make this knowledge explicit and it will require even more work to maintain it and keep it up to date. Given the problems most authors already have when they are forced to move from the "what you see is what you get" paradigm of desktop publishing to the "structured document" paradigm of XML-based Web publishing, this will not be an easy job. Having said this, we should also realize that we do not have to build it overnight: just as the current Web, we can build the Semantic Web with its Smart Style layer incrementally, by building new layers on the XML and RDF-based framework that is ready to be used now. Content-providers will start to use these new layers as soon as there are sufficiently large economic (e.g. attracting more customers by making their site accessible from new mobile devices) or legal (e.g. laws that require sites — including multimedia content — to be accessible for users with disabilities) incentives.

Acknowledgments

Many of the issues discussed were originally inspired by the W3C workshop on Device Independent Authoring in Bristol, October 2000. The research has been funded by the European ITEA/RTIPA and the Dutch Dynamo and ToKeN2000 projects. Examples are taken from a ToKeN2000 demonstrator, and all media content has been kindly provided by the Rijksmuseum in Amsterdam. Frank Cornelissen, Patrick Schmitz, Jeen Broekstra and Frank van Harmelen, and our CWI colleagues Joost Geurts, Oscar Rosell and Lloyd Rutledge provided many valuable insights.

6. REFERENCES

[1] Elisabeth André, Jochen Müller, and Thomas Rist. *WIP/PPP: Knowledge-Based Methods for Fully Automated Multimedia Authoring*. London, UK, 1996.

[2] Frederic Bes, Muriel Jourdan, and Farid Khantache. A Generic Architecture for Automated Construction of Multimedia Presentations. In *Proceedings of the International Conference on Multimedia Modeling 2001 (MMM01)*, pages 229–246, CWI, Amsterdam, The Netherlands, November 5-7, 2001.

[3] M. Bordegoni, G. Faconti, M.T. Maybury, T. Rist, S. Ruggieri, P. Trahanias, and M. Wilson. A Standard Reference Model for Intelligent Multimedia Presentation Systems. *Computer Standards & Interfaces*, 18(6-7):477–496, December 1997.

[4] Bert Bos, Håkon Wium Lie, Chris Lilley, and Ian Jacobs. Cascading Style Sheets, level 2 CSS2 Specification. W3C Recommendations are available at http://www.w3.org/TR, May 12, 1998.

[5] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: An Architecture for Storing and Querying RDF Data and Schema Information. In D. Fensel, J. Hendler, H. Lieberman, and W. Wahlster, editors, *Semantics for the WWW*. MIT Press, 2001.

[6] D.C.A. Bulterman, L. Rutledge, L. Hardman, and J. van Ossenbruggen. Supporting Adaptive and Adaptable Hypermedia Presentation Semantics. In *The 8th IFIP 2.6 Working Conference on Database Semantics (DS-8): Semantic Issues in Multimedia Systems*, Rotorua, New Zealand, 5-8 January 1999, 1999.

[7] James Clark. XSL Transformations (XSLT) Version 1.0. W3C Recommendations are available at http://www.w3.org/TR/, 16 November 1999.

[8] James Clark and Steve DeRose. XML Path Language (XPath) Version 1.0. W3C Recommendations are available at http://www.w3.org/TR/, 16 November 1999.

[9] Isabel F. Cruz, Kimberly M. James, and David C. Brown. Integrating Layout into Multimedia Data Retrieval. In *AAAI Fall Symposium on Using Layout for the Generation, Understanding, or Retrieval of Documents*, Cape Cod, Massachusetts, 1999.

[10] Jon Ferraiolo. Scalable Vector Graphics (SVG) 1.0 Specification. W3C Recommendations are available at http://www.w3.org/TR/, 4 September 2001.

[11] Ivan Herman. Follow up on SVG Linearization and Accessibility. Work in progress. Document available at http://www.w3.org/2001/svgRdf/, 19 July 2001.

[12] Yuhui Jin, Stefan Decker, and Gio Wiederhold. OntoWebber: Model-Driven Ontology-Based Web Site Management. In *Semantic Web Working Symposium (SWWS)*, Stanford University, California, USA, July 30 – August 1, 2001.

[13] Th. Kamps. *Diagram Design : A Constructive Theory*. Springer Verlag, 1999.

[14] Gregory Karvounarakis, Vassilis Christophides, Dimitris Plexousakis, and Sofia Alexaki. Querying Community Web Portals. http://www.ics.forth.gr/proj/isst/RDF/RQL/rql.html.

[15] Håkon Wium Lie and Tantek Ĉelik. Media queries. Work in progress. W3C Working Drafts are available at http://www.w3.org/TR, 17 March 2001.

[16] Guillaume Lovet. SVG Linearizer tools. Internship report and software download. Work in progress. Document available at http://www.w3.org/WAI/ER/ASVG/, 25 August 2000.

[17] Mark Oldach. *Creativity for Graphic Designers*. Narth Light Books, Cincinnati, Ohio, 1995.

[18] Lloyd Rutledge, Jacco van Ossenbruggen, Lynda Hardman, and Dick C.A. Bulterman. Structural Distinctions Between Hypermedia Storage and Presentation. In *Proceedings of ACM Multimedia*, pages 145–150. ACM Press, November 1998.

[19] Frank van Harmelen, Peter F. Patel-Schneider, and

Ian Horrocks. Reference description of the DAML+OIL (March 2001) ontology markup language. http://www.daml.org/2001/03/reference.html. Contributors: Tim Berners-Lee, Dan Brickley, Dan Connolly, Mike Dean, Stefan Decker, Pat Hayes, Jeff Heflin, Jim Hendler, Ora Lassila, Deb McGuinness, Lynn Andrea Stein, ...

[20] Jacco van Ossenbruggen. *Processing Structured Hypermedia — A Matter of Style*. PhD thesis, Vrije Universiteit, Amsterdam, The Netherlands, April 10, 2001. Also available on http://www.cwi.nl/∼jrvosse/thesis/.

[21] Jacco van Ossenbruggen, Joost Geurts, Frank Cornelissen, Lloyd Rutledge, and Lynda Hardman. Towards Second and Third Generation Web-Based Multimedia. In *The Tenth International World Wide Web Conference*, pages 479–488, Hong Kong, May 1-5, 2001. IW3C2.

[22] Jacco van Ossenbruggen, Lynda Hardman, and Lloyd Rutledge. Integrating Multimedia Characteristics in Web-based Document Languages. Technical Report INS-R0024, CWI, December 2000.

[23] Jacco van Ossenbruggen, Lynda Hardman, and Lloyd Rutledge. Hypermedia and the Semantic web: A research agenda. Technical Report INS-R0105, CWI, 2001.

[24] W3C. Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendations are available at http://www.w3.org/TR, February, 22, 1999. Editied by Ora Lassila and Ralph R. Swick.

[25] W3C. Resource Description Framework (RDF) Schema Specification 1.0. W3C Candidate Recommendations are available at http://www.w3.org/TR, 27 March 2000. Edited by Dan Brickley and R.V. Guha.

[26] W3C. Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies. Work in progress. W3C Working Drafts are available at http://www.w3.org/TR, 15 March 2001. Edited by Graham Klyne, Franklin Reynolds, Chris Woodrow and Hidetaka Ohto.

[27] W3C. Device Independence Principles. Work in progress. W3C Working Drafts are available at http://www.w3.org/TR, 18 September 2001. Edited by Roger Gimson, co-edited by Shlomit Ritz Finkelstein, Stéphane Maes and Lalitha Suryanarayana.

[28] W3C. Extensible Stylesheet Language (XSL) Version 1.0. W3C Recommendations are available at http://www.w3.org/TR/, 15 October 2001, 2001.

[29] W3C. Synchronized Multimedia Integration Language (SMIL 2.0) Specification. W3C Recommendations are available at http://www.w3.org/TR/, August 7, 2001. Edited by Aaron Cohen.

[30] W3C. XHTML 1.1 - Module-based XHTML. W3C Recommendations are available at http://www.w3.org/TR/, May 31, 2001. Edited by Murray Altheim and Shane McCarron.

[31] L. Weitzman and Kent Wittenburg. Automatic presentation of multimedia documents using relational grammars. In *Proceedings of the second ACM international conference on Multimedia '94*, pages 443–451, San Francisco, October 15 - 20, 1994.

[32] Robin Williams. *The Non-Designer's Design Book*. Peachpit Press, 1994.

[33] Wireless Application Group. WAP-174: WAG UAPROF User Agent Profile Specification, 1999.

Semantics of Links and Document Structure Discovery

John R. Punin
punin j@cs.rpi.edu
http://www.cs.rpi.edu/~ puninj
Department of Computer Science, RPI, Troy, NY 12180.

M. S. Krishnamoorthy
moorthy@cs.rpi.edu
http://www.cs.rpi.edu/~ moorthy
Department of Computer Science, RPI, Troy, NY 12180.

ABSTRACT

This paper presents a novel algorithm to discover the hierarchical document structure by classifying the links between the document pages. This link classification adds metadata to the links that can be expressed using Resource Description Framework Syntax [7]. Several well-known programs automatically generate HTML web pages from different document formats such as LaTeX, Powerpoint, Word, etc. Our interest is in the intertwined HTML web pages generated by the LaTeX2HTML program [6]. We use the web robot of the WWWPal System [11] to save the structure of the web document in a webgraph. Then the web analyzer of the system applies our algorithm to discover the semantics of the links and infer the hierarchical structure of the document.

1. INTRODUCTION

The semantic information of a document is conveyed by its logical structure. Suppose we are given a collection of URL's of a web document, along with the *up*, *previous*, and *next* links, and we are asked to determine the hierarchical structure of the document. This task can easily be accomplished using some standard traversal of the web document. However, if all the links in the web document are classified the same, the problem of finding the document's hierarchical structure is not trivial. Our hierarchical structure finding algorithm solves this nontrivial problem and discovers the logical structure.

Several well-known programs automatically generate HTML web pages from different document formats such as LaTeX, Powerpoint, Word, etc. Our interest is in the HTML web pages generated by the LaTeX2HTML program [6]. Our algorithm is also applicable to HTML documents produced by Powerpoint.

This hierarchical discovery problem is related to finding a Hamiltonian path in a webgraph. Linearization resulting from following the *next* links from the starting web page corresponds to a Hamiltonian path. Of course not all Hamiltonian paths are meaningful linearizations.

Semantic Web Workshop 2002 Hawaii, USA

In recent years, a number of software systems have been developed for the analysis of web pages, such as Mapuccino [8] [9], Microsoft Site Analyst [12] and WWWPal [11]. WWWPal is different because it can handle large graphs (called webgraphs), has a better display that uses clustering algorithms, and has a skeletal graph browser. One of WWWPal's functions is to visualize the output XGMML file from the web robot into a graph that represents the structure of the website. Several drawing algorithms have been implemented to visualize webgraphs that produce nice graph layouts. The graph visualization for web documents generated using the LaTeX2HTML program is not able to layout the hierarchical structure of the underlying web document. See Figure 1. Our algorithm discovers the structure by classifying the links and then using the semantics of the links for discovering the hierarchical structure of the web document.

In the following sections, we describe the problem statement, our algorithm, examples and the report of link classification using RDF.

2. PROBLEM STATEMENT AND ANALYSIS

2.1 Common Characters in LaTeX2HTML websites

After several LaTeX2HTML websites (web documents generated by the LaTeX2HTML program) are examined and the XGMML files are generated by a web robot, some insights for the LaTeX2HTML websites are obtained. Most of the LaTeX2HTML websites contain the following nodes/pages:

1. Table of Contents Node: Most of the LaTeX2HTML documents have this node at the top, which we can call the root. It lists all the sections the site covers much like a book's table of contents.
2. Index Node: This node functions like the book's index pages, one can go to this node and find the terminologies one is interested in. Then one can go directly to the content related to the term through the link from the Index Node.
3. Bibliography Node: This node contains the information of the author or some related materials and links.
4. Start Page Node: This is not the same as the Table of Contents Node, but it includes the same links as the Table of Contents Node. It is the initial page of the whole document.

3. ALGORITHM FOR HIERARCHICAL STRUCTURE DISCOVERY

Edges in a webgraph represent the links between web pages. These links can have a type such as: *start*, *next*, *prev*, *chapter*, etc. A group of these standard types has been recommended for HTML 4.0 [10]. When a webgraph has a hierarchical structure, we would like to visualize the webgraph as a radial tree drawing so the hierarchical structure of the document can be clearly seen. For example, the webgraph of Figure 1 and Figure 2 have the same structure. We can see that the hierarchical structure of the webgraph is not evident in Figure 1, but it is easily visible in Figure 2. The only difference between these two drawings is that in Figure 1 the type of the edges are not considered for visualization, and in Figure 2 the links of type *chapter* and *section* are visualized as tree edges. In Figure 2 we can notice that node 5 is a *chapter* of Node 1, and nodes 16 to 20 are *sections* of node 5. We can also notice that each of the children of node 1 have subtrees, and they are linked together through *next* and *prev* links. Figure 3 shows how the subtrees of node 1 are linked with the links of type *next* and *prev*.

There are many webgraph instances where the type of the links is missing. So we have to apply an algorithm to assign a type to each link of the webgraph. In this subsection we will explain a novel algorithm to classify the links of a webgraph that represents the structure of a web document. Our algorithm will only assign these four types of links: *chapter*, *section*, *next* and *prev* The following rules assign types to the edges of the webgraph:

- The children of the root node that also have a back link are considered node chapters. Therefore the edges from the root node to those children are classified as *chapter* edges.
- Our algorithm finds the subtree of the *chapter* nodes. The links between these subtrees are classified as *next* and *prev* links.
- Our algorithm orders these subtrees so there is a linear order of all nodes of the webgraph. This linear order can be considered the Hamiltonian path of the webgraph. This order can be used, for example, to make a linear print out of the web document.
- The subtrees of the webgraph are considered *chapters* of the document and each of the *chapters* have *sections*. Hence the previous rules can be applied to these *chapters* in order to find the subtree *sections*.

These rules will be applied until all edges are classified so the *chapters* and *sections* are fully resolved. For the purpose of visualization the *chapter* and *section* edges are mapped to tree edges and a radial tree drawing is applied (Figure 2).

The classification of the edges (or links) algorithm has two steps: first, the ordering of the children nodes of the root tree, and second, the traversal of the tree by retrieving the *next* node. The first task helps us to find the first child node and find the *chapter* and *section* edges; the second task linearizes the webgraph and finds the *next* and *prev* edges.

The first task is implemented in the function `get_first_child`. This function receives the root node (Start Page Node) of the subtree and returns the first of its children nodes. We

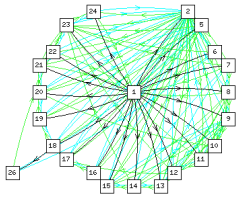

Figure 1: Radial Tree Drawing of a Web Document/webgraph

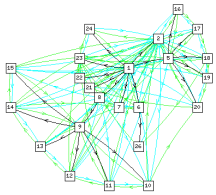

Figure 2: Typed Links Drawing of a Web Document/webgraph

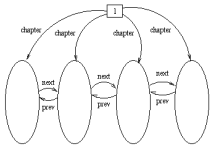

Figure 3: Subtrees of Node 1

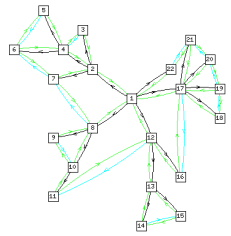

Figure 4: Example of the Typed Links Drawing

Node	Neighbor	Count	Count	Neighbor
2	8	1	—	—
8	2	3	12	1
12	17	1	8	2
17	12	2	22	1
22	17	4	—	—

Table 1: Neighbor nodes of the children of the root node and the counts

found that if we apply Breadth First Search (BFS) starting at any of the children nodes of the root node, the next child node is visited at most one time and the previous node is visited at least one time. For example, in Figure 4, the children of the root node 1 are nodes 2, 8, 12, 17 and 22. If we apply BFS starting in child node 12, node 8 is visited two times and node 17 is visited once. We can conclude that node 8 is the previous node and node 17 is the next node of node 12. With this information a table is constructed (Table 1) with the number of times that BFS visited the next and previous nodes. Using this table we can conclude what neighbors are previous and next to each children node of the root node 1. Previous nodes are the ones whose counts are greater than one and next nodes are the ones whose counts are exactly one. Table 2 shows the next and previous node for the children of the root node 1 (Figure 4). We can easily see that the first child of node 1 is node 2. Notice that the first node 2 does not have a previous node. However, we have to verify that there is a linear ordering between all the children of the root node 1 to conclude that node 2 is indeed the first node. If a first node is not found, the function `get_first_child` returns a null node and a failure error code.

The algorithm of the second step uses the `get_first_child` function to retrieve the first node of the current visited node. If first node is not found, it means that the current node is a leaf node. Hence, the next node is the neighbor node that has not been visited yet. This algorithm produces a linear

Node	Previous	Next
2	—	8
8	2	12
12	8	17
17	12	22
22	17	—

Table 2: Table of previous and next nodes for the children of the root node

```
// Linear traversal of the webgraph
int linear_traversal(PNODE root)
{
int error = 0;
PNODE next = root;
// chapter edges
classify_chapter_edges(root);
// Linear traversal
while(next && error) {
next = get_next_vertex(next, &error);
}
return error;
}
```

```
// Get next node in linear traversal
PNODE get_next_vertex(PNODE v, int *error)
{
PNODE next;
mark(v);
// get first child of children nodes
next = get_first_child(v, error);
// section edges
if(next && *error)
classify_section_edges(v);
// leaf nodes
if(!next && *error) {
next = get_unmarked_neighbor(v,error);
// next and previous edges
if(next && *error)
classify_next_prev_edges(v,next);
}
return next;
}
```

Table 3: Algorithm for classification of the edges of a webgraph

ordering of the nodes of the webgraph. The edges between the current node and the not-visited neighbors are classified as *next* and *prev* edges. The edges between the current node and its children are classified as *chapter* and *section* edges. Table 3 shows the algorithm of the second step of the webgraph traversal. Once all edges of the webgraph have been assigned a type, the *chapter* and *section* nodes are mapped to tree edges (black color), the *next* edges are mapped to forward edges (magenta color), and the *prev* edges are mapped to backward edges (green color). Afterwards, a tree or radial tree drawing can be applied to visualize the hierarchical structure the webgraph. (Figures 2 and 4).

Our algorithm fails when all the counts of the table of neighbor nodes (Table 1) are one since we will not be able to construct the table of previous and next nodes (Table 2). Without that table we cannot conclude which node is the first node. For example, Figure 5 shows a simple hierarchical webgraph where we cannot infer what node is the first node; it is either node 2 or 8. When we construct the table of neighbor nodes all counts are one as shown in Table 4.

Our algorithm is of linear time complexity as each node gets visited a constant number of times.

4. REPORT OF LINK CLASSIFICATION USING RDF

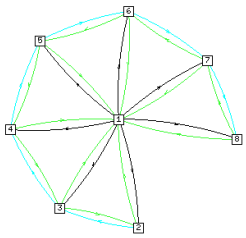

Figure 5: An example of a Web Document/webraph where the Algorithm fails.

Node	Neighbor	Count	Count	Neighbor
2	3	1	—	—
3	4	1	2	1
4	3	1	5	1
5	4	1	6	1
6	5	1	7	1
7	8	1	6	1
8	7	1	—	—

Table 4: Neighbor nodes of the children of the root node and the counts - All being equal to 1 makes the algorithm fail

The algorithm described in the previous section attaches semantics to the links and hence the web pages. One may question the usefulness of the algorithm if the semantics of hyperlinks, such as *up*, *next* and *prev* is known apriori. We envisage that explicitly specifying the links in a common semantic vocabulary will be useful to both the users and the agents.

As an example, the RDF description for node 9 (`http://cbl.leeds.ac.uk/ nikos/doc/www94/subsection3_4_1.html`) of the webgraph of the Figure 3 is as follows:

```
<rdf:RDF xmlns:rdf=''http://www.w3.org/1999/02/22-rdf-syntax-ns#''
         xmlns:rs=''http://purl.org/net/rdf/papers/sitemap#''
         xmlns:dc=''http://purl.org/dc/elements/1.1/''>
   <rdf:Description rdf:about=''http://cbl.leeds.ac.uk/~nikos/doc/www94/subsection3_4_1.html''>
      <dc:title>Common Objections to Automatic Conversion</dc:title>
      <rs:prev rdf:resource=''http://cbl.leeds.ac.uk/~nikos/doc/www94/section3_4.html''/>
      <rs:next rdf:resource=''http://cbl.leeds.ac.uk/~nikos/doc/www94/subsubsection3_4_1_1.html''/>
      <rs:section rdf:resource=''http://cbl.leeds.ac.uk/~nikos/doc/www94/subsubsection3_4_1_1.html''/>
      <rs:section rdf:resource=''http://cbl.leeds.ac.uk/~nikos/doc/www94/subsubsection3_4_1_2.html''/>
      <rs:section rdf:resource=''http://cbl.leeds.ac.uk/~nikos/doc/www94/subsubsection3_4_1_3.html''/>
      <rs:section rdf:resource=''http://cbl.leeds.ac.uk/~nikos/doc/www94/subsubsection3_4_1_4.html''/>
      <rs:section rdf:resource=''http://cbl.leeds.ac.uk/~nikos/doc/www94/subsubsection3_4_1_5.html''/>
      <rs:section rdf:resource=''http://cbl.leeds.ac.uk/~nikos/doc/www94/subsubsection3_4_1_6.html''/>
      <rs:contents rdf:resource =''http://cbl.leeds.ac.uk/~nikos/doc/www94/tableofcontents3_1.html''/>
    </rdf:Description>
</rdf:RDF>
```

The RDF triples in N-triples format [2] are:

```
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsection3_4_1.html>
<http://purl.org/dc/elements/1.1/title>
"Common Objections to Automatic Conversion" .
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsection3_4_1.html>
<http://purl.org/net/rdf/papers/sitemap#prev>
<http://cbl.leeds.ac.uk/~nikos/doc/www94/section3_4.html> .
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsection3_4_1.html>
<http://purl.org/net/rdf/papers/sitemap#next>
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsubsection3_4_1_1.html> .
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsection3_4_1.html>
<http://purl.org/net/rdf/papers/sitemap#section>
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsubsection3_4_1_1.html> .
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsection3_4_1.html>
<http://purl.org/net/rdf/papers/sitemap#section>
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsubsection3_4_1_2.html> .
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsection3_4_1.html>
<http://purl.org/net/rdf/papers/sitemap#section>
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsubsection3_4_1_3.html> .
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsection3_4_1.html>
<http://purl.org/net/rdf/papers/sitemap#section>
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsubsection3_4_1_4.html> .
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsection3_4_1.html>
<http://purl.org/net/rdf/papers/sitemap#section>
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsubsection3_4_1_5.html> .
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsection3_4_1.html>
<http://purl.org/net/rdf/papers/sitemap#section>
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsubsection3_4_1_6.html> .
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsection3_4_1.html>
<http://purl.org/net/rdf/papers/sitemap#contents>
<http://cbl.leeds.ac.uk/~nikos/doc/www94/tableofcontents3_1.html> .
```

Notice that we use the Dublin core [DC] and the RDF Sitemap vocabulary [RS].

5. EXAMPLE

We illustrate the output of our algorithm on a webgraph, obtained from visiting the LaTeX2HTML website "State of the Art Review on Hypermedia Issues And Applications" at `http://cbl.leeds.ac.uk/nikos/tmp/hypemedia/hypemedia.html`.

Figure 6 shows the radial tree drawing of the web document, and Figure 7 shows the hierarchical drawing of the same web document, where the *chapter* nodes and *section* nodes can be observed.

6. CONCLUSION

This paper is yet another important contribution to the Semantic Web [1][7][3][4] as we provide semantic classification of links and the web pages of a LaTeX2HTML website. This paper also provides a visualization tool based on our algorithm. This visualization not only draws a pretty graph, but also semantic content is imparted. Our algorithm can be extended for other types of documents.

Printing a web document, consisting of a number of web pages, is a laborious task [1], as each of the web pages has to be traversed in a linear order. This linearization is obtained automatically from the hierarchical structure order discovered by our algorithm. WWWPal uses this linearization to simplify the printing of web documents.

7. REFERENCES

[1] T. Berners-Lee, Semantic Web Area for Play: Closed World Machine. `http://www.w3.org/2000/10/swap/Overview.html`, February 2001.

[2] T. Berners-Lee, Notation 3. `http://www.w3.org/DesignIssues/Notation3.html`, April 2001.

[3] D. Brickley, RDF sitemaps and Dublin Core site summaries, `http://purl.org/net/rdf/papers/sitemap/02`), June 1999.

[4] D. Brickley, Biz/ed RDF Metadata Testbed, `http://ilrt.org/discovery/2000/08/bized-meta`, July 2001.

[5] O. De Troyer and T. Decruyenaere, "Conceptual Modelling of Web Sites for End Users," WWW Journal, Vol.3 , Issue 1, Baltzer Science Publishers, 2000.

[6] N. Drakos, "From text to hypertext: A post-hoc rationalisation of LaTeX2HTML," In Proceedings of the First World Wide Web Conference, Geneva, Switzerland, 1991.

[7] O. Lassila and R. Swick, W3C, Resource Description Framework (RDF) Model and Syntax Specification. `http://wwww.w3.org/TR/REC-rdf-syntax`, 1999.

[8] Mapuccino `http://www.ibm.com/java/mapuccino/`

[9] Y. Maarek and I. Shaul, "WebCutter: A System for Dynamic and TailorableSite Mapping," In Proceedings of WWW 6 Conference, Santa Clara, USA, 1997.

[10] D. Raggett et al, HTML 4.01 Specification, `http://www.w3.org/TR/html401/` , 1999.

[11] J. Punin and M. Krishnamoorthy, "WWWPal System - A System for Analysis and Synthesis of Web Pages", In Proceedings of the WebNet 98 Conference, Orlando, November, 1988.

[12] M. Tabini et al, Professional Site Server 3.0, Wrox Press Inc, July 1999.

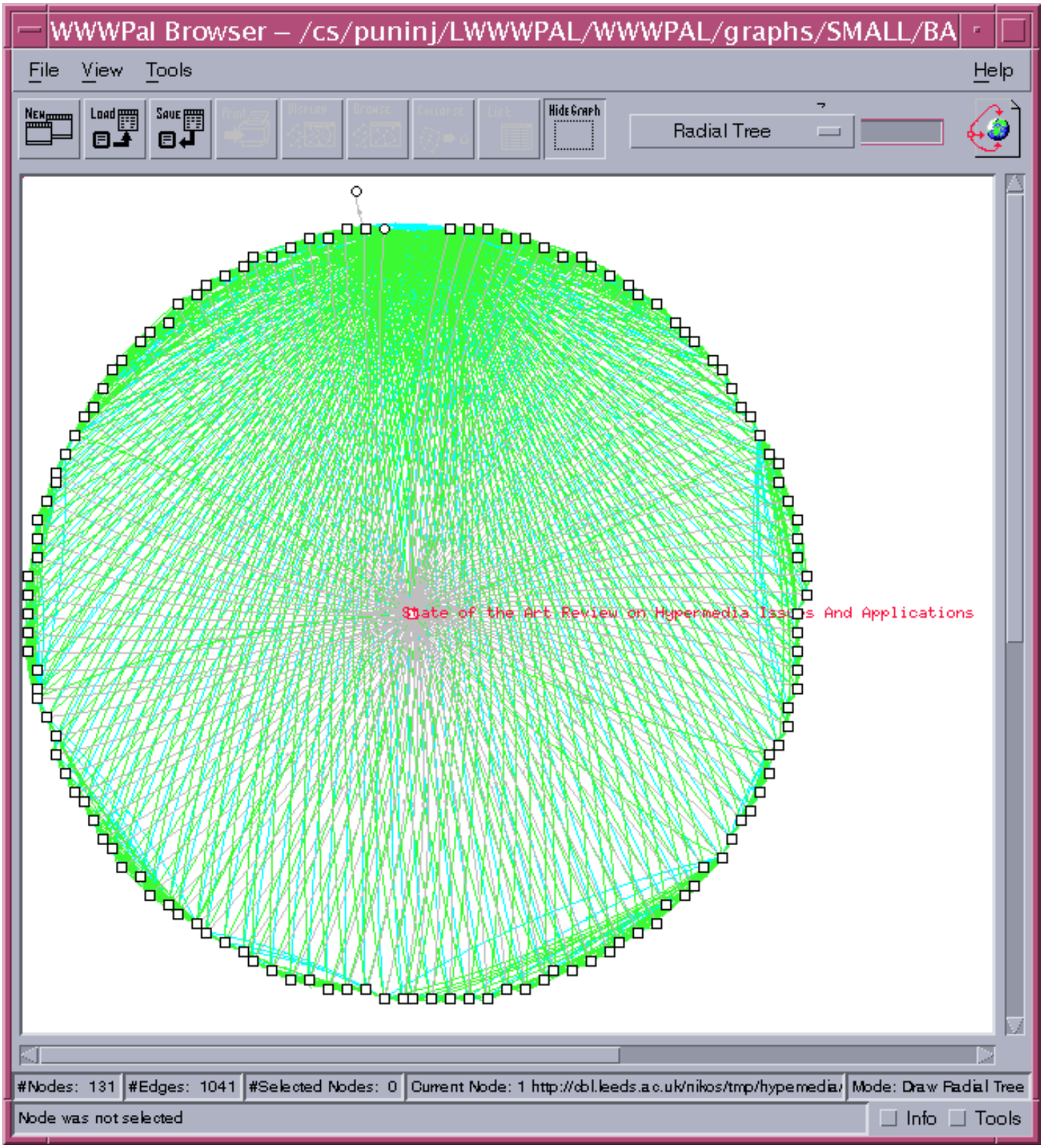

Figure 6: Radial Tree Drawing of a Web Document

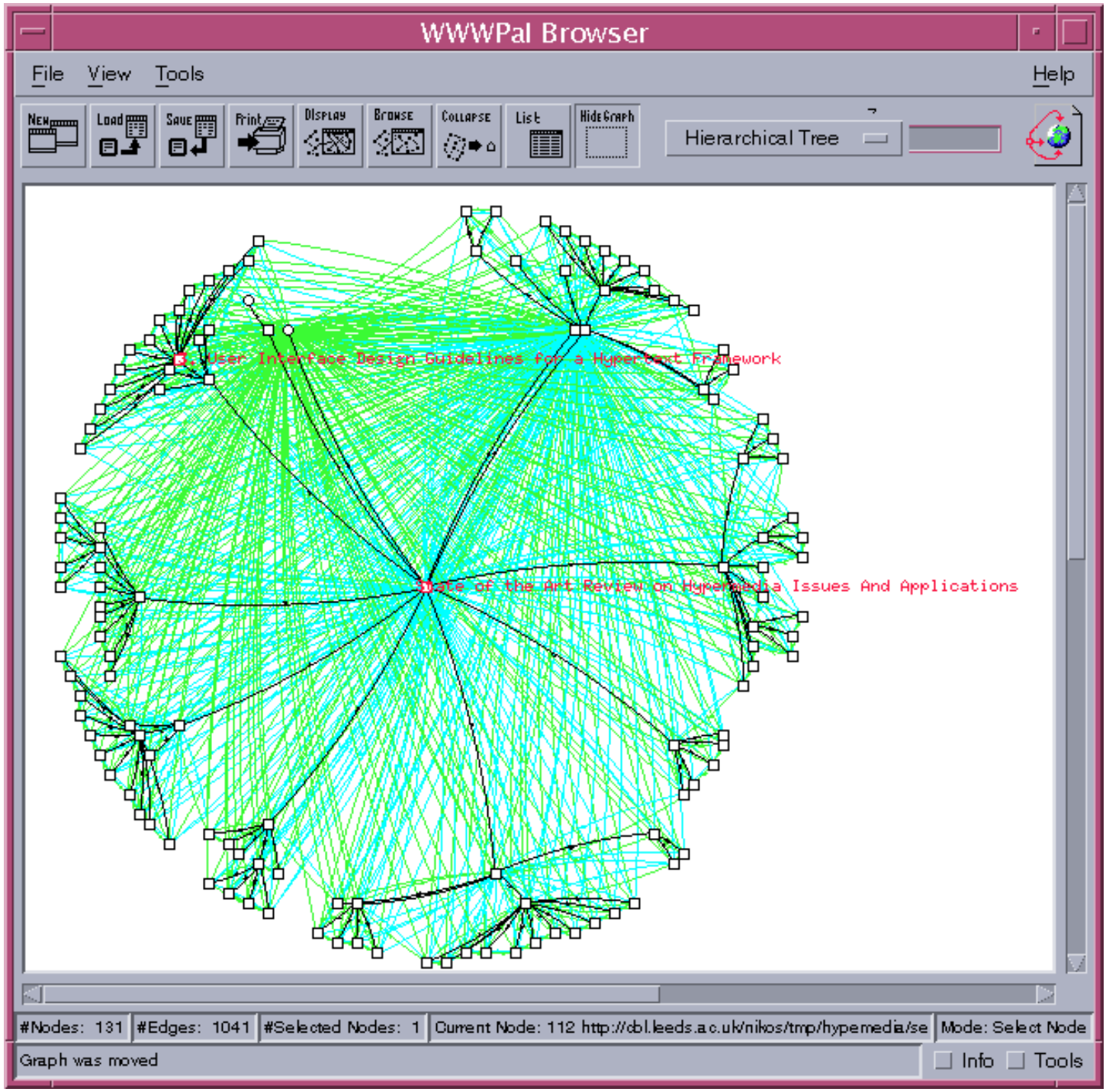

Figure 7: Hierarchical Tree Drawing of a Web Document

Haystack: A Platform for Creating, Organizing and Visualizing Information Using RDF

David Huynh
MIT Artificial Intelligence Laboratory
200 Technology Square
Cambridge, MA 02139
1 (617) 452-5041
dfhuynh@ai.mit.edu

David Karger
MIT Laboratory for Computer Science
200 Technology Square
Cambridge, MA 02139
1 (617) 258-6167
karger@theory.lcs.mit.edu

Dennis Quan
MIT Artificial Intelligence Laboratory
200 Technology Square
Cambridge, MA 02139
IBM Internet Technology Division
1 Rogers Street
Cambridge, MA 02142
1 (617) 693-4612
dquan@media.mit.edu

ABSTRACT

The Resource Definition Framework (RDF) is designed to support agent communication on the Web, but it is also suitable as a framework for modeling and storing personal information. Haystack is a personalized information repository that employs RDF in this manner. This flexible semistructured data model is appealing for several reasons. First, RDF supports ontologies created by the user and tailored to the user's needs. At the same time, system ontologies can be specified and evolved to support a variety of high-level functionalities such as flexible organization schemes, semantic querying, and collaboration. In addition, we show that RDF can be used to engineer a component architecture that gives rise to a semantically rich and uniform user interface. We demonstrate that by aggregating various types of users' data together in a homogeneous representation, we create opportunities for agents to make more informed deductions in automating tasks for users. Finally, we discuss the implementation of an RDF information store and a programming language specifically suited for manipulating RDF.

1. INTRODUCTION

The Resource Definition Framework (RDF) has been developed to provide interoperability between applications that exchange machine-understandable information on the Web [6]. In other words, RDF is well-suited for facilitating Web Services in resource discovery, cataloging, content rating, and privacy policies.

Of course, the expressive power of RDF is more far-reaching than just agent communication. We postulate that RDF can be well exploited for managing users' information. The semistructured nature of RDF lends itself well to the heterogeneous disposition of personal information corpora. In addition, since RDF provides a standard, platform-neutral means for exchanging metadata, it naturally facilitates sophisticated features such as annotation and collaboration. In this paper, we propose and demonstrate a personal information management system that employs RDF as its primary data model.

Semantic Web Workshop 2002 Hawaii, USA

1.1 Motivation

The goal of the Haystack project is to develop a tool that allows users to easily manage their documents, e-mail messages, appointments, tasks, and other information. Haystack is designed to address four specific expectations of the user.

First, the user should be allowed maximum flexibility in how he or she chooses to describe and organize his or her information. The system should allow the user to structure his or her data in the most suitable fashion as perceived by the user. Section 2 elaborates on Haystack's support for user-defined ontologies.

Second, the system should not create artificial distinctions between different types of information that would seem unnatural to the user. This point is related to the previous point in that the system should not partition a corpus simply because different programs are used to manipulate different parts of that corpus. Rather, the system should store all of the user's information in one homogeneous representation and allow the user to impose semantics that partition the data appropriately.

Third, the system should allow the user to easily manipulate and visualize his or her information in ways appropriate to the task at hand. The user interface should be aware of the context in which arbitrary information is being displayed and should present an appropriate amount of detail. We address these issues later in Section 3 where we discuss Haystack's user interface.

Fourth, the user should be able to delegate certain information processing tasks to agents. Regardless of how powerful a user interface we provide, there will still be many repetitive tasks facing users, and we feel that users will benefit from automation. The details of Haystack's agent infrastructure are given in Section 4.

1.2 Contribution

By addressing these four needs, we show that Haystack is able to use RDF to extend several profound benefits to users. First, RDF can be readily exploited to add semantics to existing information management frameworks and to serve as a *lingua franca* between different corpora. On top of this, we provide an ontology that

supports capabilities including collection-based organization, semantic categorization, and collaboration and trust management. By ontology we are referring to a vocabulary that specifies a set of classes and the properties possessed by objects of these classes. This ontology enables the user interface to present the user's information in a meaningful manner, and it also provides an abstraction on which agents can run.

Next, we show that RDF can be used to describe the means for visualizing heterogeneous data. In addition to the obvious registration metadata found in all component frameworks, RDF can be used to build user interface specification abstractions that can be directly manipulated by the user just as other metadata. This capability opens many doors for user interface engineering including the realization of a truly uniform interface.

Finally, we discuss the use of RDF for modeling imperative computational processes. We present a language called Adenine as a natural means for manipulating metadata and thus writing agents for Haystack. Adenine programs compile into an RDF representation, affording them the same ability to be annotated, distributed, and customized as other documents and information.

1.3 History

The information overload problem has become more and more evident in the past decade, driving the need for better information management tools. Several research projects have been initiated to address this issue. The Haystack project [8] [9] was started in 1997 to investigate possible solutions to this very problem. It aims to create a powerful platform for information management. Since its creation, the project has sought a data modeling framework suitable for storing and manipulating a heterogeneous corpus of metadata in parallel with a user's documents. With the introduction of RDF, a good match was found between the versatility and expressiveness of RDF and the primary need of Haystack to manage metadata. The project has recently been reincarnated to make use of RDF as its primary data model.

1.4 Related Work

There have been numerous efforts to augment the user's data with metadata. The Placeless Documents project at Xerox PARC [3] developed an architecture for storing documents based on properties specified by the user and by the system. Like Haystack, Placeless Documents supported arbitrary properties on objects and a collection mechanism for aggregating documents. It also specified in its schema access control attributes and shared properties useful for collaboration. The Placeless Documents architecture leveraged existing storage infrastructure (e.g. web servers, file systems, databases, IMAP, etc.) through a driver layer. Similarly, Haystack takes advantage of the same storage infrastructure, using URLs to identify documents.

On the surface, the main difference between Placeless' architecture and Haystack's is the adaptation of RDF as a standard for information storage and exchange. Although Haystack and Placeless share a lot of similarities in the data model layer, Haystack takes a more ambitious approach to the user interface problem. Placeless' Presto user interface focused on facilitating management of data in general using a predetermined set of interfaces. In developing Haystack, we are experimenting with ways to incorporate the customization of user interfaces into the bigger problem of personalized information management by providing a platform upon which user interfaces can be modeled and manipulated with the same facility as other metadata.

There are other systems, many in common use today, that permit arbitrary metadata annotations on files. The Windows NT file system (NTFS) supports file system-level user-definable attributes. WebDAV [2], a distributed HTTP-based content management system, also permits attributes on documents. Lotus Notes and Microsoft Exchange, two common knowledge management server solutions, both support custom attributes on objects within their databases. However, the metadata are not readily interchangeable among different environments. Further, the structure of metadata in these systems is highly constrained and makes the expression of complex relationships between objects difficult. For example, these systems do not have first class support for making assertions about predicates, making it difficult for the user interface and agents to analyze data conforming to a foreign ontology dynamically.

The Semantic Web project at the World Wide Web Consortium (W3C), like Haystack, is using RDF to address these issues of interchangeability [4]. The focus of the Semantic Web effort is to proliferate RDF-formatted metadata throughout the Internet in much the same fashion that HTML has been proliferated by the popularity of web browsers. By building agents that are capable of consuming RDF, data from multiple sources can be combined in ways that are presently impractical. The simplest examples involve resolving scheduling problems between different systems running different calendaring servers but both speaking RDF. A more complex example is one where a potential car buyer can make automated comparisons of different cars showcased on vendors' web sites because the car data is in RDF. Haystack is designed to work within the framework of the Semantic Web. However, the focus is on aggregating data from users' lives as well as from the Semantic Web into a personalized repository.

2. DESCRIBING AND ORGANIZING HETEROGENEOUS INFORMATION

In this section we examine strategies for managing a heterogeneous corpus. First we examine how users can define objects using their own ontology. Then we discuss one means for aggregating objects—the collection—and how it is used in Haystack to help users organize their information.

2.1 Personalized Ontologies

One of Haystack's objectives is to facilitate the use of an ontology for organizing, manipulating and retrieving personal information. Some classes will be defined in the system ontology, such as those used for describing queries and collections of objects. Also, some properties, such as title, language, and description, will be defined by standard ontologies such as the Dublin Core. Other classes and properties can be defined by the user to suit his or her own organization method.

On the surface, the Haystack model may not seem very different from those of current systems. Programs such as Lotus Notes and Microsoft Outlook support classes such as e-mail message, contact, and appointment and provide default properties, including subject, from, and date. However, whereas the focus of these products is in providing an efficient and user-friendly means for maintaining objects with these standardized schemata, Haystack attempts to facilitate the entry of data using the user's own class definitions. This functionality is typically found in relational database products, where users first specify a structured schema that describes the data they are entering before populating the table. In reality however, schema design is usually left for

database and system administrators, who circulate their designs toward end users of Notes or Outlook.

This "one size fits all" approach for schema design is far from perfect. End users are a fundamentally diverse populace, and people often have their own ideas of what attributes to store for particular classes of objects. A good example is an address book. Some people only care about storing one or two phone numbers and a mailing address, while sales managers may be concerned with a breakdown of all past contact with a customer as well as important dates in the customer's life, such as his or her birthday. The current approach of making more and more fields built-in to an address book product is problematic. Software adopting this approach is often overloading to the user who just wants a simple address book, yet perhaps not functional enough for the sales manager. Using a personal database such as Microsoft Access or FileMaker Pro only aggravates this problem, since users are forced to rebuild their address books from generic templates and generic data types.

To solve this mismatch problem, we must examine means for describing per-user customization. Technologies such as RDF provide flexible data frameworks upon which customized schema definitions and metadata can be specified. RDF's data model encourages the creation of custom vocabularies for describing the relationships between different objects. Furthermore, RDF's XML syntax makes these personalized vocabulary specifications portable between different systems. This will be important for allowing agents to enhance the user's information, as is discussed later.

The challenge exists in how to bring this ability to customize a personal information store to the end user who has no experience with database administration. To accomplish this, we have devised tools for helping people manipulate unstructured, semi-structured, and structured data, abstracting the details of schema management from end users. These tools are built upon a flexible, semi-structured RDF data store that Haystack uses to manage users' information according to ontologies they choose.

We generalize the problem of editing data based on arbitrary ontologies by providing a generic metadata editor. This editor takes advantage of the RDF Schema [7] data within Haystack's RDF store in order to present the user with a useful interface to their data. Future versions will allow users to assign arbitrary properties (not just those specified by the schema) to objects by simply typing the name of the property and its value. In this way users need not be conscientious about schemata, and incidental properties specific to one object and not to the class can be entered.

A customized view of an object will often provide a better interface to the user than a generic tool, when one is available. To support this we provide a sophisticated platform for describing these customizations in our prototype user interface tool called Ozone, discussed in Section 3.

In addition to editing properties of specific objects, it is often useful to the user to be able to manipulate the relationship between objects. A graph editor allows users to see a collection of objects and add and/or remove relationships between these objects. This idea has been popularized in tools such as Microsoft Visio, where structures such as flow charts, organization charts, and business processes can be modeled in a similar fashion. Further, tools for drawing semantic diagrams by making connections between concepts have become available. The Haystack graph editor provides these functionalities to the user but records this data in RDF, making the semantic diagrams true representations of "live" data.

2.2 Classifying Information

While users may be interested in customizing how their contact information is stored in their address books, some abstractions we argue are best defined by the base system. This prevents the user from being too bogged down with details of semantic modeling, while providing the user with out-of-the-box functionality. Here we discuss one of these key classes, Collection.

A big problem with many document management systems, including paper-based ones, is the inability to conveniently file documents in more than one category. Although hierarchical folders are a useful and efficient means for storing documents, the hierarchical folder system presents challenges to users who attempt to use it to categorize documents. Does a document named "Network Admin Department Budget for 2001" belong in the "Budget" folder, the "2001" folder, or the "Network Admin" folder? Supposing an individual finds justification for placing it in just one of these folders, it is very possible that other users may have different views on classification and expect the document to be in a different folder. It may also be the case that some days a user will be interested in a time-based classification and other days a department-based classification.

Simply supporting files being in more than one folder at once is not sufficient. Commonly used modern operating environments such as Windows and MacOS already provide mechanisms (called "shortcuts" and "aliases" respectively) for placing objects in more than one folder. On UNIX systems users can create hard links to files in more than one directory at once. However, we find relatively little use of these features for simultaneously classifying documents into multiple categories.

We postulate that this is because the user interface does not encourage simultaneous classification. How many programs can be found whose file save feature prompts the user for all the possible directories into which to place a file? Many users place their files into a single directory because they are not willing to expend the effort to classify files. Of the fraction that are willing, there is yet a smaller fraction who would be willing to save their files in one place, then go to the shell to create the hard links into the other directories.

Collections, like folders, are aggregations of objects; an object may be a member of more than one collection, unlike folders, whose interface encourages a strict containment relationship between folders and objects. This flexible system for grouping objects together is an important tool for allowing users to organize objects in any way they choose.

Throughout the Haystack user interface, we provide simple facilities for specifying membership in more than one collection. Of course, we support direct manipulation schemes such as drag and drop between collections. However, as noted earlier, the system must itself facilitate the placement of objects in multiple collections in order to be useful. For example, Haystack generates collections to store the results of queries. Whereas in some systems, such as Microsoft Outlook, first class membership in multiple collections is only supported when the collection is a search result set, this functionality is supported naturally within

Haystack. Still, we envision the greatest source of multiple classification will be from agents automatically categorizing documents for the user.

3. SEMANTIC USER INTERFACE

In addition to modeling data, RDF is used as the medium for specifying Haystack's user interface and how Haystack presents the user's data. Haystack's prototype user interface, named Ozone, uses a custom component architecture, and the user interface is constructed dynamically at runtime based on metadata. In this section, we introduce this metadata-based component architecture, show how it helps construct a uniform user interface, and describe the benefits such an interface might bring.

3.1 Component Architecture

The Ozone user interface is constructed from a conglomeration of *parts*. An Ozone part is a code-backed component that can contribute to the user interface in some way. Attributes of each part are provided in metadata. In particular, the part's implementation, the types of data it can render, and the functionality it provides are described.

3.1.1 Types of Parts

There are four types of parts: layout parts, informative parts, decorative parts, and view parts. Layout parts are responsible for segmenting the visual display into distinct spaces and for positioning other parts in these spaces. Informative parts present textual and graphical information to the user. Decorative parts provide decorations such as white margins, line dividers, text spacing, list separators, etc. Finally, view parts use layout parts, informative parts, and decorative parts as building blocks in constructing a unified view of a single object to the user.

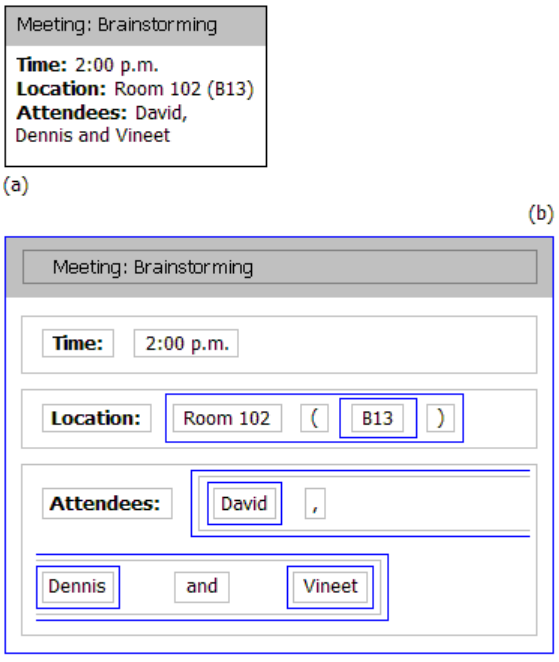

Figure 1. Example of different types of parts working together

Figure 1 shows an example of how the various types of parts work together to present a meeting. Part (a) of the figure shows the end result while part (b) shows the internal wiring. The view part responsible for displaying the meeting employs a vertically splitting layout part to partition the display into two rows: the top row embeds an informative part that renders the title of the meeting; the bottom row contains the details of the meeting. The bottom row in turn contains a stacking layout part that stacks the three fields "Time," "Location," and "Attendees" vertically.

The "Location" field consists of a decorative part that renders the label "Location:" and a view part that displays the room where the meeting is held. Note that because the room is a separate entity, the meeting view part does not attempt to present the room itself but rather employs another view part specialized to present the room. (The room is a separate entity because it has been modeled as a resource rather than as a literal property of the meeting.) The room view part includes an informative part to display the room's title "Room 102," two decorative parts to show the parentheses, and yet another view part to display the building where the room is located.

The "Attendees" field consists of a decorative part that renders the label "Attendees:" and a view part that shows the collection of attendees. The collection view part uses a list layout that positions the collection members sequentially, with decorative parts showing comma and "and" separators in between. The collection members are rendered by their own appropriate view parts.

Note that each view part is responsible for displaying exactly one semantic entity. In Figure 1, there are seven distinct semantic entities: the meeting, the room, the building, the attendee collection, and the three individual attendees. If a semantic entity is related to other semantic entities, the view part for that entity may incidentally embed view parts for the other entities. The parent view part determines the appropriate type of each child view part to embed, so that the nested presentation looks pleasing. For instance, a small parent view part embeds only small child view parts.

3.1.2 Part Metadata

Given a semantic entity to present, Ozone queries the RDF store for the part suitable for displaying the entity. Figure 2 shows an example of the metadata that links the entity to the suitable part.

In order to display the hs:favorites entity, Ozone queries for any view instance associated with the entity through the hs:view predicate.[1] If no view instance is found, Ozone determines the type of the entity (in this case, hs:Collection) and the view class corresponding to that type (ozone:ListView). Ozone then instantiates a unique resource that serves as a view instance for the hs:favorites entity and asserts that the view instance is of that view class type. The view instance will be persisted thereafter and it will serve to store custom settings applied by the user while viewing the corresponding semantic entity. In this example, such settings include the columns and their order in the list view. Each type of view instances persists its settings in its own custom ontology.

[1] The hs: prefix denotes a URI belonging to the Haystack namespace. Note that the idea of views is inherent to the Haystack ontology, whereas the view parts used to display them are inherent to Ozone.

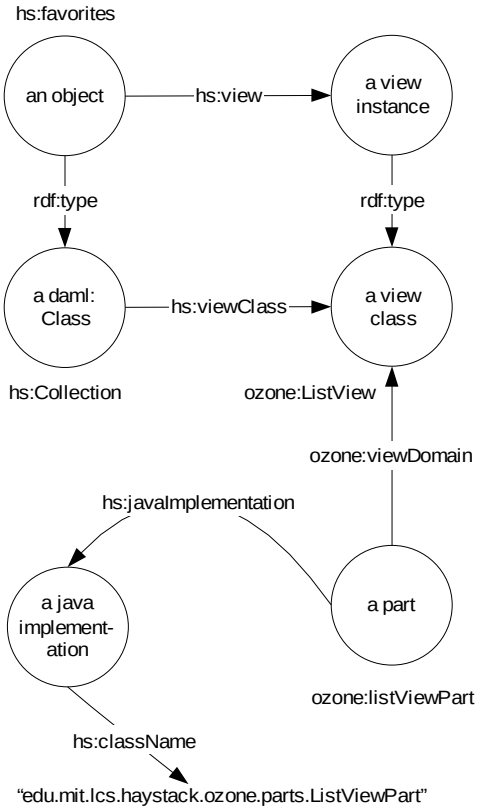

Figure 2. Part metadata example

Once a view instance exists and its view class is known, Ozone queries for a part whose view domain (ozone:viewDomain) is the class of the given view instance. In this example, the part ozone:listViewPart is found. Ozone then finds the implementation of this part, instantiates the corresponding Java class, and initializes it with the semantic entity.

3.1.3 Benefits

The ability to encode the user interface in metadata and then render the interface from that metadata is appealing for a number of reasons. First, it allows the component architecture to construct the user interface dynamically at run-time; any change made to the metadata can be applied immediately. This is not possible with conventional user interface programming models for which user interface layouts are compiled into binary resources or code and loaded at runtime. Any change to such layouts requires recompilation unless the code is specifically parameterized to accept runtime customization. Even in rapid application development tools like Microsoft Visual Basic in which user interfaces can be built by drag and drop operations, there are two distinct edit and runtime modes. The user is only allowed to interact with the application in runtime mode when the user interface is already fixed and unchangeable. Skinnable applications also have a similar limitation. Skins are made by skin designers and then published to users. The users can select which skins to apply to the applications, but they cannot customize the skins themselves. Again, there are two distinct design and use modes that deny the users the power of customizing the user interfaces themselves. Likewise, web pages have two edit and view modes: in view mode, the user cannot make modifications. Our Ozone interface architecture imposes no such modes and allows the user to make changes to the user interface at runtime.

Note that user interface changes performed by the user are high-level: they are to programmers' user interface work as interior design is to carpentry. In other words, customizing the user interface is akin to editing a word processing document or manipulating a spreadsheet. The user is allowed to arrange parts and apply visual themes over them; all the necessary "carpentry work" is handled automatically by Ozone. Since arguably there is no universal interface that suits the needs of every user, this ability to customize one's user interface is desirable. In fact, such personalization features have been explored in simple forms on several portal web sites like http://my.yahoo.com. We would like to bring personalization to our information management platform. In addition to arranging pre-installed parts, the user is offered to select new parts from a part repository and drag them to desired locations in the Ozone interface.

One might argue that similar component architectures and personalization engines have been implemented without the need for RDF. In fact, anything implemented using RDF can be done with custom formats. However, what RDF offers is a unified and extensible framework much needed in the presence of several incompatible custom formats.

The second benefit of user interface metadata is that, like any other type of data, the user interface metadata can be saved, published, and shared. The ability to publish user interface metadata is particularly attractive. Consider a scenario in which a new employee enters a company with a large intranet. This intranet offers countless useful resources to the new employee, but because of its sheer volume, it is intimidating to get used to. Fortunately, other employees have over time collected a selection of Ozone parts that provide access to the most useful features of the intranet. The new employee can simply make use of these parts as a starting point. These parts are brought into the employee's Haystack and tailored based on his or her preferences. These parts can interact with the employee's Haystack and perform automatic customization that makes Haystack much more powerful than a static web page listing a set of useful links to the intranet.

User interfaces, hence, can be evolved by the users for themselves as their needs emerge and coalesce. Users with different needs tailor their interfaces differently. Those with similar needs share their interface layouts. This philosophy relieves the interface designers from the impossible task of making universal interfaces that satisfy both expert and novice users. Instead, we provide tools for the user to tailor his or her own user interface. In addition, by providing an ontology describing user interface interactions, such interactions can be tracked automatically and agents can apply machine learning algorithms to better fit the user interface to the user's implicit needs and preferences.

The third benefit of user interface metadata is that the user interface designer is encouraged to think semantically as he or she encodes the interface in metadata. Since the interface is rendered automatically by the component architecture based on a unified set of semantics, the barrier to creating user interfaces is much lowered. By removing the burden of fiddling with the interface "until it works" or "so it looks nice," we encourage the designer to think at the level of the user's semantics rather than at the level of how the user interface is syntactically constructed. Such

abstraction leads to consistency, the most advocated property of user interface [10].

Finally, because the user interface is modeled in RDF just as the user's data is, the tools offered by Ozone for allowing the user to manipulate his or her data are the same as those used when manipulating the interface itself. For example, the mechanism for changing a contact's name is also used for changing the caption of a button. The mechanism for changing the font in an email message body can be used to change the font in which all labels are displayed. In fact, the mechanism for looking up a word in the body of a document is made available for looking up a menu command that one does not understand. This uniform fashion in which all things can be manipulated makes the interface of Ozone consistent and hence, natural and powerful.

3.2 Uniform User Interface

Using the power of the component architecture, we explore the concept of a *uniform user interface* [10]. In such an interface, any two user interface elements that look semantically identical to the user afford the same set of actions regardless of context. For instance, a contact name shown in the "From" column for an email message in a list view (Figure 3) should expose the same actions as the same contact name shown in the "From" text field in the email message window (Figure 4). In Microsoft Outlook XP and other existing email clients, those two elements provide almost entirely different sets of actions. The former element is a dead text string painted as part of the whole list view item representing the email message. Right-clicking on it is equivalent to right-clicking anywhere in that list view item. The same context menu for the whole message is always shown regardless of the right-click location. The latter element is a control by itself. It represents a contact object and shows the context menu applicable to that object when right-clicked. To the user, both elements represent the same contact object and should give the same context menu. (Uniformity in this case implies modelessness.)

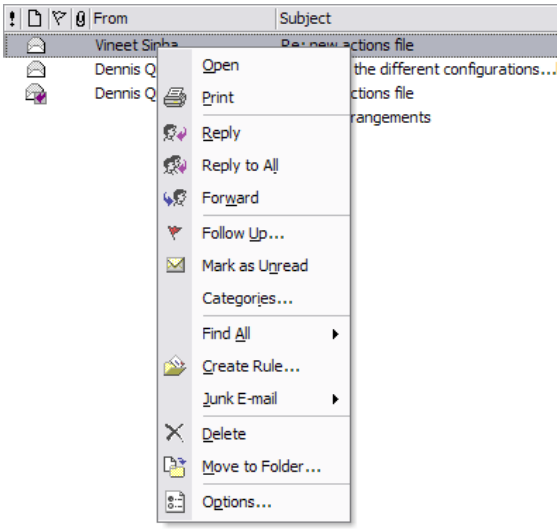

Figure 3. Actions for a contact name in a list view (Microsoft Outlook XP)

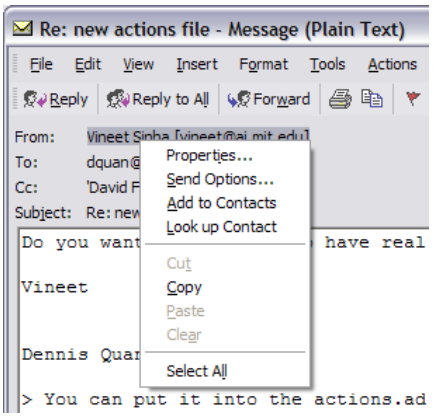

Figure 4. Actions for a contact name in the email compose window (Microsoft Outlook XP)

Context menus have been adopted by most operating systems as the mechanism to query for actions applicable to the indicated object. However, context menus are rarely used consistently throughout any user interface. Some user interface elements afford context menus while others do not. Furthermore, some context menus do not expose all possible actions that the user associates semantically with their corresponding objects. Their inconsistency and incompleteness make context menus less powerful, and hence, less useful and less widely adopted by users than they should be. We aim to fix such inconsistency and incompleteness by providing context menus for all user interface elements and by systematically constructing the menus from metadata.[2] We believe that this uniformity will make the interface much easier to use.

In order to construct context menus from metadata, we first note that every pixel on the screen corresponds to a particular Ozone part that renders that pixel. If that part is a view part, there is a corresponding semantic entity that it is responsible for displaying. That view part can be contained in other outer view parts. All of these view parts together specify a collection of semantic entities that underlie the pixel. We postulate that one of these semantic entities is the thing with which the user wants to interact. To construct a context menu when that pixel is right-clicked, we simply list all actions applicable to those semantic entities.

Like semantic entities, Ozone parts can also afford actions. For instance, the informative part that displays the text "Vineet Sinha" in Figure 3 allows the user to copy its text. The layout part that shows the messages in a list view format allows the user to reorder the columns.

Figure 5 gives a sample context menu that will be implemented for an upcoming release. The menu is divided into several sections, each listing the commands for a particular semantic entity or Ozone part. The email author entity named "Vineet Sinha" is given the first section. The email message entity is given the next section. Finally the text label used to display the contact's name is given the third section. Commands for other semantic

[2] Context menus are not widely adopted also because they are not currently discoverable. Once discovered, they are very memorable. We propose labeling the right mouse button "Show Commands" to make context menus more discoverable.

entities and parts can be accessed through the "More…" item at the bottom of the menu. This is only an example of how context menus can be constructed. The exact order of the sections will be optimized by user study and feedback.

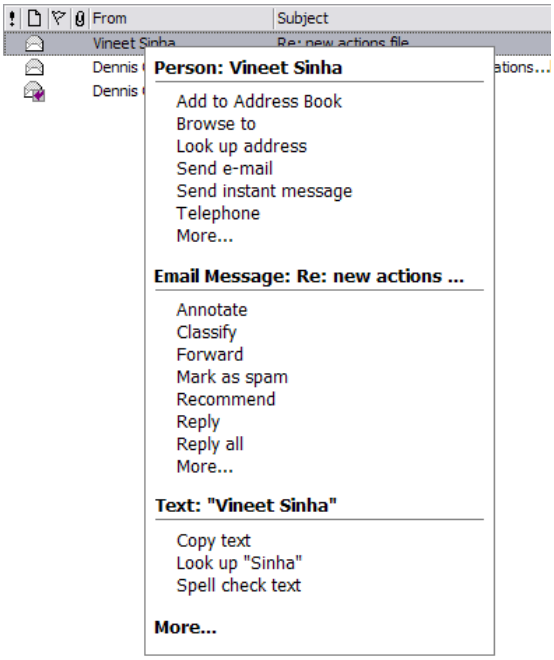

Figure 5. Sample context menu

Of note in Figure 5 are some capabilities not provided in existing email clients. In particular, the "Text" section in the menu offers ways to copy, look up, and spell-check an otherwise dead piece of text. In other email clients, only text inside email bodies can be spell-checked. One can also imagine the usefulness of spell-checking file names [10] and email subjects.

It is arguable that when a user interface element is associated with several semantic entities and parts, its context menu will be overloaded. We believe that with proper modeling of prioritization of commands and in-depth user study, we can heuristically select the most useful commands to list in the menu. Other commands are still accessible through "More…" links. Further, because menu commands are simply members of the collection of all actions applicable to some semantic entity, the user can use the mechanism provided by Ozone for browsing and searching collections to locate particular menu commands. In existing software, menu commands can only be found by visual scans.

4. AGENT INFRASTRUCTURE

We now turn our attention to agents, which play an important role in not only improving the user experience with regards to keeping information organized, but also in performing tedious tasks or well-defined processes for the user. We also describe some underlying infrastructure needed to make writing and using agents in Haystack efficient and secure.

4.1 Agents

In the past, programs that aggregated data from multiple sources, such as mail merge or customer relationship management, had to be capable of speaking numerous protocols with different back-ends to generate their results. With a rich corpus of information such as that present in a user's Haystack, the possibility for automation becomes significant because agents can now be written against a single unified abstraction. Furthermore, agents can be written to help users deal with information overload by extracting key information from e-mail messages and other documents and presenting the user with summaries.

As we alluded to earlier, collections can be maintained automatically by agents. Modern information retrieval algorithms are capable of grouping documents by similarity or other metrics, and previous work has found these automatic classifications to be useful in many situations. Additionally, users can build collections prescriptively by making a query. An agent, armed with a specification of what a user is looking for, can create a collection from the results of a query, and it can watch for new data entering the system that matches the query.

For example, agents can automatically filter a user's e-mail for documents that appear to fit in one or more collections defined by the user, such as "Website Project" or "Letters from Mom". Because membership in collections is not one-to-one, this classification can occur even while the message remains in the user's inbox.

Agents are used in Haystack to automatically retrieve and process information from various sources, such as e-mail, calendars, the World Wide Web, etc. Haystack includes agents that retrieve e-mail from POP3 servers, extract plaintext from HTML pages, generate text summaries, perform text-based classification, download RSS subscriptions on a regular basis, fulfill queries, and interface with the file system and LDAP servers.

The core agents are mostly written in Java, but some are written in Python. We utilize an RDF ontology derived from WSDL [5] for describing the interfaces to agents as well as for noting which server processes hosts which agents. As a consequence, we are able to support different protocols for communicating between agents, from simply passing in-process Java objects around to using HTTP-based RPC mechanisms such as HTTP POST and SOAP [1].

4.2 Belief

When multiple agents are used to generate the same information, issues arise as to how to deal with conflicts. For instance, if one agent is tasked with determining the due date of a document by using natural language processing and another agent does the same by extracting the first date from a document, which is to be believed when there is a conflict? In instances such as this, it is important that information be tagged with authorship metadata so the user can make an informed choice of which statement to choose.

To accomplish this we discuss a part of the system ontology that is used for describing attributes about actual statements themselves, such as who asserted them and when they were asserted. Under the premise that only three values, namely subject, predicate, and object, are required to describe statements in our model, it is possible to give statements identifiers and to assert an author and creation time to the original statement. In fact, the RDF model prescribes that in order to make statements about statements, the referent statement must be reified into a

resource and assigned a URI, and the referring statements can then use the reified resource in the subject or object field.

This use of reification brings up a subtle issue concerning RDF. In a document containing RDF, it is assumed that all statements are asserted to be true by the author. In order to make a statement about another statement that the author does not necessarily believe is true, the target statement must exist only in reified form. In essence, the author is binding a name to a specific statement with a certain subject, predicate, and object, but is not asserting the statement to be true, only instead asserting other properties about that statement using the name.

Keeping track of various levels of trustworthiness is important in a system that contains statements made by numerous independent agents, as well as information from users' colleagues, friends, family, solicitors, and clients. In order to maintain metadata on the statements themselves in an information store, one solution is to have the information store become a "neutral party", recording who said what and when those things were said, but not asserting their truth. This is accomplished by having all statements made by parties other than the information store reified. (An alternative is to have one entity—perhaps the user--be at the same trust level as the data store. However, this results in statements made by the user being handled in one fashion and those made by others (which have been reified) handled in a different fashion. For simplicity of implementation, we keep the data store neutral.)

Once we have a system for recording statement metadata, we can examine issues of retraction, denial, and expiration of assertions, *i.e.*, statements asserted by specific parties. Consider an example where an agent is responsible for generating the title property for web pages. Some web pages, such as those whose contents are updated daily, have titles that change constantly. Often users want to be able to locate pages based on whatever it is they remember about the page. One approach for handling constant mutations in the information store is to allow agents to delete a previous assertion and to replace it with an up-to-date version. However, it would be powerful to allow users to make queries of the form "Show me all web pages that had the title *Tips for Maintaining Your Car* at some point in time." By allowing agents to retract their assertions, queries can still be made to retrieve past or obsolete information because this information is not deleted. Additionally, this system permits users to override an assertion made by an agent by denying the assertion, yet retains the denied assertion for future reference.

In a system such as this where multiple parties and agents provide information, we are often concerned with impersonation and forgery. To solve these problems, we propose supporting digitally signed RDF. The digital signature permits the information store to determine and verify the author of statements with certainty. In an ideal system, users and agents sign all RDF they produce with assigned digital signatures. However, the W3C is still working on the details of supporting signed RDF at the statement level, and the implementation of a digital signature system is beyond the scope of this project. For our current prototype, identifier strings are used in place of true signatures.

4.3 Adenine

In a system such as Haystack, a sizeable amount of code is devoted to creation and manipulation of RDF-encoded metadata. We observed early on that the development of a language that facilitated the types of operations we frequently perform with RDF would greatly increase our productivity. As a result, we have created Adenine. An example snippet of Adenine code is given in Figure 6.

The motivation for creating this language is twofold. The first key feature is making the language's syntax support the data model. Introducing the RDF data model into a standard object-oriented language is fairly straightforward; after all, object-oriented languages were designed specifically to be extensible in this fashion. Normally, one creates a class library to support the required objects. However, more advanced manipulation paradigms specific to an object model begin to tax the syntax of the language. In languages such as C++, C#, and Python, operator overloading allows programmers to reuse built-in operators for manipulating objects, but one is restricted to the existing syntax of the language; one cannot easily construct new syntactic structures. In Java, operator overloading is not supported, and this results in verbose APIs being created for any object oriented system. Arguably, this verbosity can be said to improve the readability of code.

On the other hand, lack of syntactic support for a specific object model can be a hindrance to rapid development. Programs can end up being three times as long as necessary because of the verbose syntactic structures used. This is the reason behind the popularity of domain-specific programming languages, such as those used in Matlab, Macromedia Director, etc. Adenine is such a language. It includes native support for RDF data types and makes it easy to interact with RDF containers and services.

The other key feature of Adenine is its ability to be compiled into RDF. The benefits of this capability can be classified as portability and extensibility. Since 1996, p-code virtual machine execution models have resurged as a result of Java's popularity. Their key benefit has been portability, enabling interpretation of software written for these platforms on vastly different computing environments. In essence, p-code is a set of instructions written to a portable, predetermined, and byte-encoded ontology.

```
# Prefixes for simplifying input of URIs
@prefix : <urn:test-namespace:>

:ImportantMethod rdf:type rdfs:Class

method :expandDerivedClasses ; \
rdf:type :ImportantMethod ; \
rdfs:comment \
"x rdf:type y, y rdfs:subClassOf z => x rdf:type z"
          # Perform query
          # First parameter is the query specification
          # Second is a list of the variables to return,
          # in order
          = data (query {
                    ?x rdf:type ?y
                    ?y rdfs:subClassOf ?z
          } (List ?x ?z))

          # Assert base class types
          for x in data
                    # Here, x[0] refers to ?x
                    # and x[1] refers to ?z
                    add { x[0] rdf:type x[1] }
```

Figure 6. Sample Adenine code

Adenine takes the p-code concept one step further by making the ontology explicit and extensible and by replacing byte codes with RDF. Instead of dealing with the syntactic issue of introducing byte codes for new instructions and semantics, Adenine takes advantage of RDF's ability to extend the directed "object code"

graph with new predicate types. One recent example of a system that uses metadata-extensible languages is Microsoft's Common Language Runtime (CLR). In a language such as C#, developer-defined attributes can be placed on methods, classes, and fields to declare metadata ranging from thread safety to serializability. Compare this to Java, where serializability was introduced only through the creation of a new keyword called `transient`. The keyword approach requires knowledge of these extensions by the compiler; the attributes approach delegates this knowledge to the runtime and makes the language truly extensible. In Adenine, RDF assertions can be applied to any statement.

These two features make Adenine very similar to Lisp, in that both support open-ended data models and both blur the distinction between data and code. However, there are some significant differences. The most superficial difference is that Adenine's syntax and semantics are especially well-suited to manipulating RDF data. Adenine is mostly statically scoped, but has dynamic variables that address the current RDF containers from which existing statements are queried and to which new statements are written. Adenine's runtime model is also better adapted to being run off of an RDF container. Unlike most modern languages, Adenine supports two types of program state: in-memory, as is with most programming languages, and RDF container-based. Adenine in effect supports two kinds of closures, one being an in-memory closure as is in Lisp, and the other being persistent in an RDF container. This affords the developer more explicit control over the persistence model for Adenine programs and makes it possible for agents written in Adenine to be distributed.

The syntax of Adenine resembles a combination of Python and Lisp, whereas the data types resemble Notation3 [11]. As in Python, tabs denote lexical block structure. Backslashes indicate a continuation of the current line onto the next line. Curly braces ({}) surround sets of RDF statements, and identifiers can use namespace prefixes (e.g. rdf:type) as shorthand for entering full URIs, which are encoded within angle brackets (<>). Literals are enclosed within double quotes.

Adenine is an imperative language, and as such contains standard constructs such as functions, for loops, arrays, and objects. Function calls resemble Lisp syntax in that they are enclosed in parentheses and do not use commas to separate parameters. Arrays are indexed with square brackets as they are in Python or Java. Also, because the Adenine interpreter is written in Java, Adenine code can call methods and access fields of Java objects using the dot operator, as is done in Java or Python. The execution model is quite similar to that of Java and Python in that an in-memory environment is used to store variables; in particular, execution state is *not* represented in RDF. Values in Adenine are represented as Java objects in the underlying system.

Adenine methods are functions that are named by URI and are compiled into RDF. To execute these functions, the Adenine interpreter is passed the URI of the method to be run and the parameters to pass to it. The interpreter then constructs an initial in-memory environment binding standard names to built-in functions and executes the code one instruction at a time. Because methods are simply resources of type adenine:Method, one can also specify other metadata for methods. In the example given, an rdfs:comment is declared and the method is given an additional type, and these assertions will be entered directly into the RDF container that receives the compiled Adenine code.

The top level of an Adenine file is used for data and method declarations and cannot contain executable code. This is because Adenine is in essence an alternate syntax for RDF. Within method declarations, however, is code that is compiled into RDF; hence, methods are like syntactic sugar for the equivalent Adenine RDF "bytecode".

Development on Adenine is ongoing, and Adenine is being used as a platform for testing new ideas in writing RDF-manipulating agents.

5. DATA STORAGE

5.1 RDF Store

Throughout this paper we have emphasized the notion of storing and describing all metadata in RDF. It is the job of the RDF store to manage this metadata. We provide two implementations of the RDF store in Haystack. The first is one built on top of a conventional relational database utilizing a JDBC interface. We have adopted HSQL, an in-process JDBC-compatible database written in Java. However, early experiments showed that for the small but frequent queries we were performing to render Ozone user interfaces, the RDF store was overloaded by the fixed marshalling and query parsing costs. Switching to a larger scale commercial database appears to result in worse performance because of the socket connection layer that is added in the process.

To solve these problems we developed an in-process RDF database written in C++ (we use JNI to connect it to the rest of our Java code base). By making it specifically suited to RDF, we were able to optimize the most heavily used features of the RDF store while eliminating a lot of the marshalling and parsing costs. However, we acknowledge this to be a temporary solution, and in the long term we would prefer to find a database that is well-suited to the types of small queries that Haystack performs.

5.2 Storing Unstructured Content

It is important for us to address how Haystack interacts with unstructured data in the existing world. Today, URLs are used to represent files, documents, images, web pages, newsgroup messages, and other content accessible on a file system or over the World Wide Web. The infrastructure for supporting distributed storage has been highly developed over the past decades. With the advent of technologies such as XML Namespaces and RDF, a larger class of identifiers called URIs subsumed URLs. Initially, RDF provided a means for annotating web content. Web pages, identified by URL, could be referred to in RDF statements in the subject field, and this connected the metadata given in RDF to the content retrievable by the URL. This is a powerful notion because it makes use of the existing storage infrastructure.

However, with more and more content being described in RDF, the question naturally arises: why not store content in RDF? While this is certainly possible by our initial assumption that RDF can describe anything, we argue this is not the best solution for a couple of reasons. First, storing content in RDF would be incompatible with existing infrastructure. Second, leveraging existing infrastructure is more efficient; in particular, using file I/O and web protocols to retrieve files is more efficient than using XML encoding.

Hence, we do not require that existing unstructured content be stored as RDF. On the contrary, we believe it makes sense to store

some of the user's unstructured data using existing technology. In our prototype, we provide storage providers based on HTTP 1.1 and standard file I/O. This means that storing the content of a resource in Haystack can be performed with HTTP PUT, and retrieving the content of a resource can be performed with HTTP GET, analogously to how other resources' contents (e.g., web pages) are retrieved. Our ontology uses the Content class and its derivatives, HTTPContent, FilesystemContent, and LiteralContent to abstract the storage of unstructured information.

6. PUTTING IT TOGETHER

At this point, we have described ontologies for personal information management and user interfaces, as well as an agent infrastructure and a data storage layer. In order to gain a fuller understanding of how these components work together, we illustrate an example interaction between the user and Haystack.

Figure 7 shows the user's home page, which is displayed when Ozone is first started. Like a portal, the Ozone home page brings together in one screen information important to the user. This information is maintained by agents working in the background. The actual presentation of this information is decoupled from the agents and is the responsibility of Ozone view parts. For instance, the home page displays the user's incoming documents collection, which is managed by the Incoming Agent. When messages arrive, the Incoming Agent may decide to enter them into the incoming documents collection. Similarly, when read messages have been in the incoming documents collection for some period of time, the Incoming Agent may decide to remove them. These mutations to the incoming documents collection are automatically detected by the collection view part sitting on the home page; the view part updates the display accordingly. One can envision the Incoming Agent taking on more intelligent behaviors in the future, such as moving a message deduced to be important but yet unread to the top of the collection.

As mentioned earlier, strings of text on the screen corresponding to appointments, news articles, or e-mail messages are not merely dead pixels. Instead, users can manipulate them with context menus and drag and drop them between different areas of the screen. For example, one can imagine dragging an e-mail from the incoming documents view to the calendar in order to set up an appointment. Because the underlying semantic object is connected to the visual representation, the calendar view part can

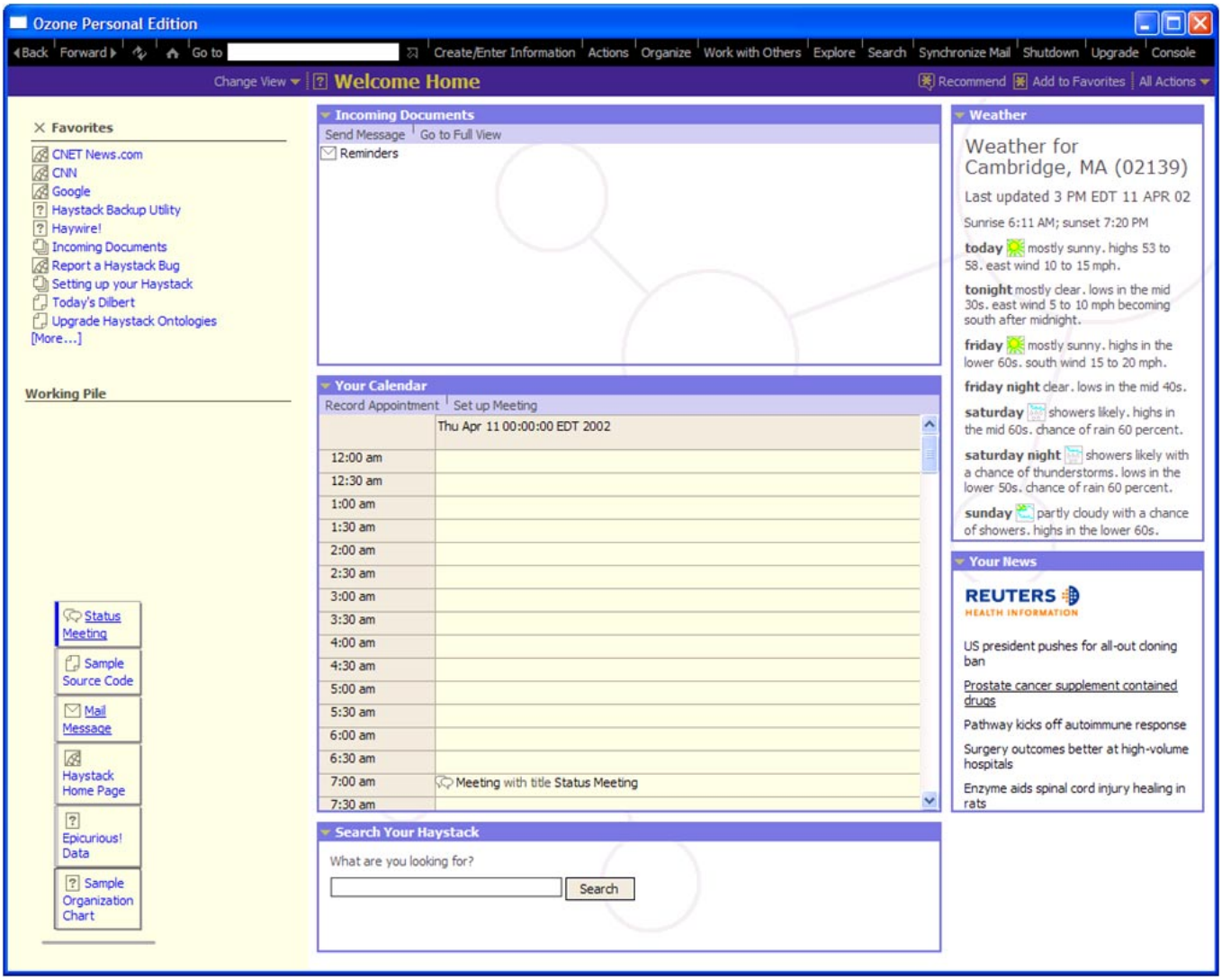

Figure 7. Ozone screenshot

intelligently determine the correct response to the drop operation.

By removing the burden of user interface rendering from the agents, the software designers are encouraged to enrich the agents with more capabilities. One can imagine prolific collaboration between different agents in the Haystack system. For instance, upon retrieving the weather forecast for today, the Weather Agent can notify the Calendar Agent of the grave possibility of a snow storm approaching; the Calendar Agent in turn can attempt to reschedule the user's appointments appropriately. In other systems, especially portals, the focus of a weather agent would be on rendering the weather as HTML, not interacting with other agents to maximize end user benefit.

The news applet displays news downloaded from Resource Site Summary (RSS) feeds of interest to the user. The RSS Agent downloads news on a periodic basis and incorporates the RSS files (which are RDF) into the user's corpus. To take advantage of the collection view part for displaying news, another agent translates the news from the RSS ontology into the Haystack collection ontology. In the future it will be possible to have another agent filter the RSS feeds for the particular articles thought to be most interesting to the user.

Furthermore, the layout of the entire home page and all of its customizations is described in metadata. As with other objects, this layout can be annotated, categorized, and sent to others.

7. FUTURE WORK

Haystack provides a great platform for organizing and manipulating users' information. In this section we touch upon two topics we are currently investigating that build new abstractions on top of the data model discussed above.

7.1 Collaboration

Enabling users to work together, exchange information, and communicate has become an absolutely essential feature of modern information management tools. The focus of current off-the-shelf products has been on e-mail and newsgroup-style discussions. However, the addition of rich metadata manipulation facilities creates many possibilities for Haystack in fostering collaboration.

First, Haystack encourages users to have individualized ontologies, so converting between these ontologies when exchanging data will need to be examined. Agents can be instructed in the relationships between different ontologies and can perform conversion automatically. As an alternative one can imagine an ontological search engine that is consulted whenever a user enters data. This way users end up using the same ontologies to describe similarly-structured data.

Second, security issues arise when sharing data. Support for belief networks will need to be expanded to allow users to distinguish their own information from information obtained from others. Access control and privacy will need to be examined to allow users to feel comfortable about storing information in Haystack.

Finally, metadata describing individual users' preferences towards certain topics and documents can be used and exchanged to enable collaborative filtering. Sites such as epinions.com promote user feedback and subjective analysis of merchandise, publications, and web sites. Instead of going to a separate site, users' Haystacks can aggregate this data and, by utilizing the belief network, present users with suggestions.

7.2 Organization Schemes

We have started to investigate the many ways in which people organize their personal information in physical form, such as bookcases and piles. We believe that each method of organization has different advantages and disadvantages in various situations. In light of this, we propose to support several virtual organization schemes simultaneously, such that the user can choose the appropriate organization scheme to use in each situation. Different schemes act like different lenses on the same corpus of information. We will provide agents that help the user create and maintain these organization schemes.

8. REFERENCES

[1] Box, D., Ehnebuske, D., Kavivaya, G., et al. *SOAP: Simple Object Access Protocol*. http://msdn.microsoft.com/library/en-us/dnsoapsp/html/soapspec.asp.

[2] Goland, Y., Whitehead, E., Faizi, A., Carter, S., and Jensen, D. *HTTP Extensions for Distributed Authoring – WEBDAV*. http://asg.web.cmu.edu/rfc/rfc2518.html.

[3] Dourish, P., Edwards, W.K., et al. "Extending Document Management Systems with User-Specific Active Properties." *ACM Transactions on Information Systems*, vol. 18, no. 2, April 2000, pages 140–170.

[4] Berners-Lee, T., Hendler, J., and Lassila, O. "The Semantic Web." *Scientific American*, May 2001.

[5] Christensen, E., Cubera, F., Meredith, G., and Weerawarana, S. *Web Services Description Language (WSDL) 1.1*. http://www.w3.org/TR/wsdl.

[6] *Resource Description Framework (RDF) Model and Syntax Specification.* http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.

[7] *Resource Description Framework (RDF) Schema Specification.* http://www.w3.org/TR/1998/WD-rdf-schema/.

[8] Adar, E., Karger, D.R., and Stein, L. "Haystack: Per-User Information Environments" in *1999 Conference on Information and Knowledge Management*.

[9] Karger, D and Stein, L. "Haystack: Per-User Information Environments", February 21, 1997.

[10] Raskin, J. "The Humane Interface." Addison-Wesley, 2000.

[11] Berners-Lee, T. *Primer: Getting into RDF & Semantic Web using N3*. http://www.w3.org/2000/10/swap/Primer.html.